TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Echtzeitsysteme und Robotik

Non-Linear Control Strategies for Musculoskeletal Robots

Michael Jäntsch

Vollständiger Abdruck der von der Fakultät der Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ. Prof. Dr.-Ing. Alin Albu-Schäffer
Prüfer der Dissertation:	1. Univ. Prof. Dr.-Ing. habil. Alois Knoll
	2. Dr. Guido Herrmann, Reader (University of Bristol, UK)

Die Dissertation wurde am 09. Oktober 2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 05. Februar 2014 angenommen.

ABSTRACT

Recently, focus has shifted to more human-friendly robots, especially in the field of service or rehabilitation robotics, where research aims at bringing robots into increasingly unstructured environments. Here, humans still outperform robots in almost every aspect. One way towards this goal is to mimic more and more of the mechanisms of the human musculoskeletal system. These so called musculoskeletal robots incorporate compliant antagonistic actuation in a biologically inspired skeleton. Furthermore, complex joints like spherical joints and multi-articular muscles are realized. In the last decade impressive musculoskeletal robots have been developed, where the focus was mainly on constructing working prototypes. However, the field of controlling such robots has made very little progress, so far. Reasons for this are manifold, starting from unavailable sensor modalities, due to novel skeletal structures, to the difficulties in modeling the interaction of the muscles with the skeleton.

In this work, (1) a generic model for the class of musculoskeletal robots was derived by extending standard models for robot dynamics, (2) several non-linear control strategies were developed and (3) a robotic arm was constructed to evaluate the novel control approaches. Non-linear controllers derived in this work include feedback linearization, which is also the main control technique used for tendon-driven robots and was therefore extended to be applied to the characteristics of musculoskeletal robots. Furthermore, a novel control law, based on Dynamic Surface Control (DSC), which is an extension to *backstepping*, was developed. In contrast to previously used techniques, the systematic approach of *backstepping* provides a method to obtain a cascade of controllers without neglecting the dynamics of the low-level control laws, hence providing provably stable closed-loop behavior. This scheme was further extended by adaptive neural network control to compensate for unmodeled dynamics such as friction. The dynamic model, incorporated by the different controllers utilizes a standard rigid body model of the skeleton which was extended to support spherical joints. In the presence of complex joint types and muscles that wrap around skeletal structures, analytic models of the mapping between the joint and the muscle space are extremely complex to obtain. To overcome this issue machine learning techniques were applied to attain the muscle kinematics.

The different control laws were evaluated on a physical robotic platform which was specifically developed to cover the control challenges of the class of musculoskeletal robots. Hence it comprises a spherical shoulder joint and a revolute elbow joint. Actuation was realized by several uni-articular muscles and a bi-articular muscle spanning both joints. Results obtained on this platform show that it is possible to achieve stable closed-loop behavior, where the novel control laws developed in this work yield improved trajectory tracking performance, compared to existing approaches.

ZUSAMMENFASSUNG

In jüngster Zeit hat sich der Forschungsschwerpunkt im Bereich der Service- und Rehabilitationsrobotik auf Roboter verschoben, die auch in zunehmend unstrukturierten Umgebungen und in enger Zusammenarbeit mit dem Menschen eingesetzt werden können. In diesen Szenarien sind Menschen nach wie vor in der Lage, Roboter in jeglicher Hinsicht zu übertreffen. Eine Möglichkeit diesbezüglich Fortschritte zu erzielen, ist die konsequente Nachahmung des menschlichen Bewegungsapparats. Diese sogenannten musculoskeletalen Roboter bestehen aus einem System aus antagonistisch angeordneten Muskeln und einem biologisch inspirierten Skelett, wobei sowohl komplexe Gelenktypen, wie z.B. Kugelgelenke, als auch multiartikulare Muskeln realisiert werden. Obwohl in den letzten Jahren einige beeindruckende musculokeletale Roboter konstruiert wurden, lag der Fokus bisher hauptsächlich auf der Realisierung von betriebsbereiten Prototypen, während im Bereich der Regelung bisher nur wenige Fortschritte erzielt werden konnten. Dies liegt zum einen an der Verfügbarkeit der notwendigen Sensoren für die neuartigen Gelenkstrukturen, als auch an der Modellierungsproblematik der Muskelkinematik, d.h. des Zusammenspiels zwischen Muskeln und Skelett.

In dieser Arbeit wurden (1) ein generisches Dynamikmodell für die Klasse der musculoskeletalen Roboter hergeleitet, indem die Standardmodelle aus der klassischen Robotik erweitert wurden, (2) mehrere nichtlineare Regler entwickelt und (3) ein Roboterarm konstruiert, auf dem die nichtlinearen Regler evaluiert wurden. Bei den nichtlinearen Reglern kamen sowohl Feedback-Linearisierung, als auch Dynamic Surface Control (DSC), welches auf *Backstepping* basiert, zum Einsatz. Während Ersteres die bisherige Lösung für die Regelung von sehnengetriebenen Systemen darstellt und lediglich auf die spezifischen Problemstellung von musculoskeletalen Robotern zugeschnitten wurde, stellt Letzteres einen neuen Lösungsansatz dar. Der DSC Ansatz wurde zusätzlich um Adaptive Neuronale Netze zur Kompensation von Reibungseffekten erweitert. Die Problematik der Modellierung der Muskelkinematik in solchen Systemen wurde mithilfe von Algorithmen aus dem Bereich des maschinellen Lernens gelöst. Dies ist speziell für den Einsatz in Systemen mit komplexen Gelenktypen und Muskeln, die das Skelett umschließen, sinnvoll, da analytische Modelle in diesem Fall extrem komplex werden können.

Die unterschiedlichen Regelungsansätze wurden auf Basis eines realen Robotersystems evaluiert, welcher explizit dafür entwickelt wurde, die Regelungsproblematik der musculoskeletalen Roboter abzudecken. Aus diesem Grund besteht dieser aus einem Kugelgelenk für die Schulter und einem Drehgelenk für den Ellbogen und wird mit mehreren uniartikularen und einem biartikularen Muskel angetrieben. Die erzielten Evaluierungsergebnisse zeigen, dass die entwickelten Regelungsalgorithmen in der Lage sind das Gesamtsystem zu stabilisieren und die Regelgüte gegenüber existierenden Algorithmen zu verbessern.

Trust should be the basis for all our moral training.

— Robert Baden-Powell

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Prof. Alois Knoll for the opportunity to work on this interesting and challenging project and supporting this thesis. My sincere thanks go to Dr. Guido Herrmann for numerous helpful discussions and his suggestions for improvement.

I am very grateful to have had Steffen Wittmeier as an office mate. Without the many discussions we had during those years, the work presented here would not have led as far. Special thanks go also to Dr. Konstantinos Dalamgkidis for guidance and helpful comments. I would also like to thank all the other colleagues at the Robotics & Embedded Systems Group, as well as the Eccerobot and the Myorobotics team for interesting topics and the time we shared.

I thank my family for all the help and constant encouragement, as well as my brother Martin for proof-reading. Finally, special thanks go to Yvonne for supporting me, especially during those last months.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

AD	Analog-to-Digital
AI	Artificial Intelligence
ANN	Artificial Neural Network
CAD	Computer Aided Design
CAN	Controller Area Network
CFC	Computed Force Control
CRONOS	Conscious Robot (Or Near Offer) System
DC	Direct Current
DoF	Degrees of Freedom
DSC	Dynamic Surface Control
Eccerobot	Embodied Cognition in a Compliantly Engineered Robot
ECU	Electronic Control Unit
EDS	Eccerobot Design Study
EMF	back Electro-Magnetic Force
EMG	Electromyography
EPH	Equilibrium Point Hypothesis
FFW	Feed-Forward
FFW-NN	Feed-Forward Neural Network
FSM	Finite State Machine
FSR	Force Sensitive Resistor
IDMH	Internal Dynamics Model Hypothesis
JSK	Jouhou System Kougaku Laboratory of the University of Tokyo
LP	Linear in the Parameters

MLP	Multilayer Perceptron
MSE	Mean Square Error
NBR	Nitrile Butein Rubber
ODE	Ordinary Differential Equation
OFC	Optimal Feedback Control
PA-2200	Polyamid 2200
P	Proportional
PD	Proportional-Derivative
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PWM	Pulse-Width Modulation
RBF	Radial Basis Function
RBFN	Radial Basis Function Network
RLS	Recursive Least Squares
RMSE	Root Mean Square Error
SEA	Series Elastic Actuator
SLS	Selective Laser Sintering
USB	Universal Serial Bus
UUB	Uniformly Ultimately Bounded
XML	Extensible Markup Language

SYMBOLS

${}^{a}p_{b}$	Position of coordinate frame b in coordinate frame a
${}^{a}x$	Position vector x in coordinate frame a
${}^{a}R_{b}$	Rotation of coordinate frame b in coordinate frame a
${}^{a}T_{b}$	Transformation matrix of a coordinate frame b in coordinate frame a
${}^{0}T_{i}, {}^{0}R_{i}$	Transformation and rotation matrix of the i^{th} joint frame
${}^{0}T_{i'}, {}^{0}R_{i'}$	Transformation and rotation matrix of the i^{th} moving joint frame
$\vec{\omega}$	3×1 vector of rotational velocity
Q	Quaternion
$S(\cdot)$	3×3 skew symmetric matrix
I_{o}	$o \times o$ identity matrix
N	number of joints (counting joints instead of DoF)
n	number of joint coordinates (skeletal DoF)
m	number of muscle coordinates
q	$n \times 1$ vector of joint coordinates
Δq	$n \times 1$ vector of joint coordinate errors
τ	$n \times 1$ vector of joint torques
F	6×1 vector of operational space forces
$J(q)$	$6 \times n$ joint Jacobian
$H(q)$	$n \times n$ joint space mass matrix
$C(q, \dot{q})$	$n \times n$ joint space matrix of coriolis terms
$\tau_{G}(q)$	$n \times 1$ joint space gravitational torques
f	$m \times 1$ vector of muscle forces
θ	$m \times 1$ vector of motor coordinates
$l(q)$	$m \times 1$ vector of muscle lengths with respect to joint angles
i_{A}	$m \times 1$ vector of anchor currents
v_{A}	$m \times 1$ vector of anchor voltages
J_{T}	Combined motor and gearbox moment of inertia $J_{T} = J_{M} + J_{G}$
K	$m \times m$ diagonal matrix of spring stiffnesses
D	$m \times m$ diagonal matrix of damping coefficients
R_{S}	$m \times m$ diagonal matrix of the spindle radii
$L(q)$	$m \times n$ muscle Jacobian

PUBLICATIONS

Some of the work presented in this thesis has been published:

1. M. Jäntsch, S. Wittmeier, and A. Knoll. Distributed control for an anthropomimetic robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 5466–5471, Oct. 2010. doi: 10.1109/IROS.2010.5651169

2. M. Jäntsch, C. Schmaler, S. Wittmeier, K. Dalamagkidis, and A. Knoll. A scalable Joint-Space Controller for Musculoskeletal Robots with Spherical Joints. In *Proc. IEEE International Conference on Robotics and Biomimetics ROBIO*, pages 2211–2216, Dec. 2011. doi: 10.1109/ROBIO.2011.6181620

3. M. Jäntsch, S. Wittmeier, K. Dalamagkidis, and A. Knoll. Computed Muscle Control for an Anthropomimetic Elbow Joint. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 2192–2197, Oct. 2012. doi: 10.1109/IROS.2012.6385851

4. M. Jäntsch, S. Wittmeier, K. Dalamagkidis, A. Panos, F. Volkart, and A. Knoll. Anthrob – A Printed Anthropomimetic Robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013

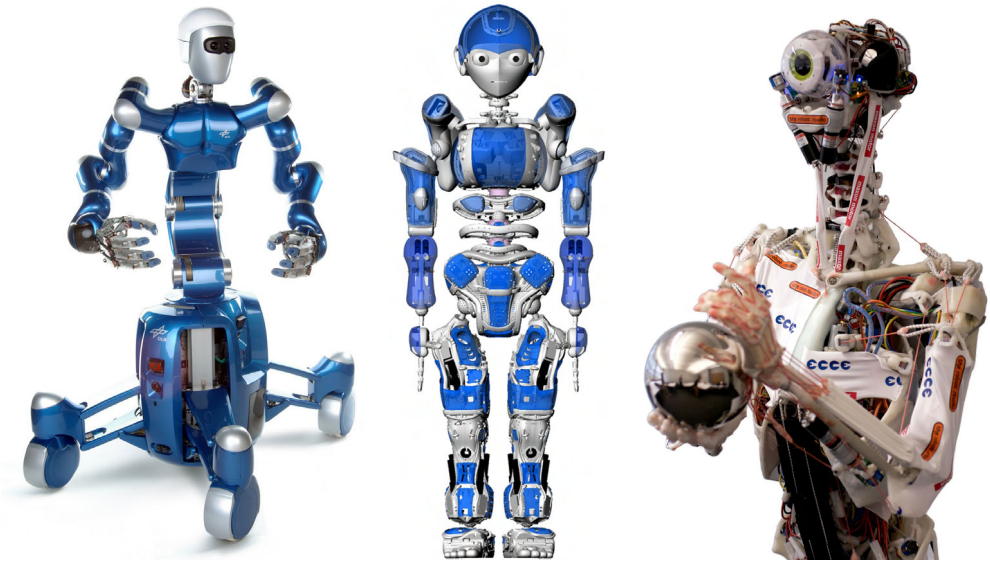
1 INTRODUCTION

1.1 BACKGROUND

Up to now, robots have been mainly used in factory environments, where industrial robots offer extremely high precision and repeatability for tasks in which trajectories can be planned ahead. In these cases it was the unrivaled performance which led to the high increase in robot utilization in large production sites, where major parts of the manufacturing process can be automated and robots can work unhindered by the presence of humans.

Recently, the focus has shifted to more human-friendly robots, especially in the field of service or rehabilitation robotics, but also when physical human-robot interaction for manufacturing tasks is considered. Service robots will work together with humans in our future homes, to serve in a variety of assignments, ranging from entertainment to difficult or tedious tasks such as helping individuals with special needs or fetching and delivering objects [116]. This field is identified as a large market for robotics in Europe, but the robots of today have shortcomings when unstructured environments are considered. Here, humans still outperform robots in almost every aspect. This is due to the uncertainty that comes with these environments which cannot be sufficiently perceived and modeled and therefore leads to errors in task execution. Compared to production sites, the challenges lie both in the structure of the objects, which robots will be interacting with, but also in the unpredictability of other agents like humans.

In these cases stiff robots are very likely to damage themselves or their surroundings. This problem can be diminished but never truly solved by probabilistic planning algorithms and ever better sensors. The chance of collisions between robots and humans can be diminished but never be fully eliminated, risking serious injuries [39]. The technology of soft robots is widely considered as the method of bringing robots into our modern homes and factories [6]. These robots are characterized by additional compliance in the actuation mechanisms which can, on the one hand, be achieved by so called active compliance, utilizing rich sensor feedback and a superimposed impedance control [42]. Hence, the robot reacts with the impedance defined by the controller. Passive compliance, on the other hand, is realized in hardware by mechanically adding an elastic element between the drive and the joint. Here, we distinguish between traditionally actuated robots, where drives are located in the joints, and mechanisms which mimic the biological concept of antagonistic actuation. These soft tendon-driven systems, feature opposing actuators with an elastic element in the tendon which offer better intrinsic robustness to kinematic errors and are capable of using all motors for changing the stiffness, as well as driving the



(a) (b) (c)

Fig. 1.1: Compliant Humanoid Robots. (a) The actively compliant robot Justin [6] by the DLR, (b) Kojiro [85] by the JSK and (c) The anthropomimetic robot EDS [76] by The Robot Studio.

joints [6]. A subcategory of these robots is identified, which do not only mimic the type of actuation but feature also biologically inspired skeletons and muscles and are hence called musculoskeletal robots.

Up to now, most soft robots are realized by a rigid structure and actively compliant joints (see Fig. 1.1a). However, this approach is limited by sensor bandwidth and control frequency. Especially in cases where the robot collides with rigid objects at high speeds, this limitation poses a huge threat to the robot joints, since control is not fast enough to react to sudden impacts [39]. The advantages of using passively compliant actuators for robotics can be clearly identified as the ability to absorb energy in the elastic elements. This can be exploited (1) to safely work in unstructured environments, as collisions with objects or possible human co-workers do not necessarily lead to human injuries and/or the overloading of the robot's joints and (2) by actively storing energy. The latter could be utilized in periodic tasks, like walking or playing the drums, by cyclically storing and releasing the energy in the elastic elements, which leads to higher energy efficiency. Furthermore, the elasticity can be exploited by pre-loading. When performing fast dynamic movements, like throwing a ball, the human body is able to store energy by co-contracting muscles. This yields the possibility of releasing this energy at will and with much higher forces and speeds as would have been possible otherwise [6].

By employing tendon-driven actuation, the actuators can be placed with more freedom, to e. g. improve the weight distribution of the robot. Therefore, this technique has been identified as a possible solution in high speed manipulation tasks or for the design

of robotic hands. In contrast, musculoskeletal robots combine this type of actuation with the advantages of soft robots and the biomimetic idea of understanding by building [102]. The *Anthropomimetic Principle* [43] proposes to not only copy the outside of the human body, but also the internal mechanisms, in order to build humanoid robots that do not only appear to have the outer shape of a human, but also its internal dynamic properties. This includes the skeleton, as well as muscles, tendons and ligaments. Hence these anthropomimetic robots form a subcategory of the larger class of musculoskeletal robots, which do not necessarily mimic humans but also other biological systems by employing artificial muscles. Currently, there are only a few robots that copy the human internal structure in the sense of the *Anthropomimetic Principle*. Examples are the robots Kenta, Kotaro, Kojiro, Kenzoh and Kenshiro (see Fig. 1.1b and [67, 83–85, 90]), built by the Jouhou System Kougaku Laboratory of the University of Tokyo (JSK), as well as the ECCEs, developed within the EU-funded project Embodied Cognition in a Compliantly Engineered Robot (Eccerobot) (see Fig. 1.1c and [143]). The latter emerged from the project Conscious Robot (Or Near Offer) System (CRONOS) [43]. By employing this principle, a completely new type of robot emerges, whose dynamic properties are very similar to the ones of humans. The *Anthropomimetic Principle* yields the possibility to build humanoid robots that can work in unstructured environments due to their soft nature. Additionally the system dynamics of such robots facilitate the interaction with humans. At the same time they incorporate the advantages of passively compliant and tendon-driven robots.

1.2 MOTIVATION

Even though, research of the last years has produced extremely impressive robots that were able to mimic more and more details of the sensory-motor system of biological creatures, the field of controlling such robots has made very little progress, so far. While demonstrations of these robots show mere Feed-Forward (FFW) control, the existing feedback control techniques from the field of tendon-driven control [50, 99, 113] have failed to scale to more complex structures. This is due to the fact that musculoskeletal robots in general exhibit several characteristics that are usually not present in engineered tendon-driven systems. These include (1) the integration of joints with multiple Degrees of Freedom (DoF) like spherical joints, (2) the additional elastic elements in the tendon paths, (3) multi-articular muscles and (4) so called ill-defined muscle kinematics. In most tendon-driven robots the lever arms of the tendons remain constant with the joint angles due to pulleys that guide the tendons. In musculoskeletal robots this is not the case, however, and lever arms change while the system moves. Furthermore, the complex skeletal structures irrevocably lead to muscles that collide with the bones in some configurations. Those ill-defined muscle kinematics are extremely complex to model analytically. However, specifically the existence of those challenging structures accounts to a large degree for the dynamics of biological systems. For instance, it has been shown that bi-articular

muscles are responsible for the synchronization of several joints in explosive movements, like jumping [134].

Modeling and generating physiologically consistent trajectories for simulated musculoskeletal systems is at the same time ongoing research in biomechanics, where the goal is to compute "muscle excitation patterns that produce coordinated movements" [131]. In biomechanics the motivation is to analyze the role of different muscles in certain movements, where only a few states, like joint angles or excitation patterns of some muscles, can be measured. While not exactly solving the control problem, the definition of biomechanical models can provide the missing link between the existing controllers for tendon-driven robots and the controllers to be developed for musculoskeletal robots.

The construction of anthropomimetic robots is of great interest not only to the field of robotics, but also for neuroscientists and biomechanicists. There have been several hypotheses, including the Equilibrium Point Hypothesis (EPH) [25] and the Internal Dynamics Model Hypothesis (IDMH) [59], of how humans are able to control their bodies, which even extensive discussions and experiments could not prove or refute. By developing algorithms for the exact problem of controlling human-like dynamic structures, possibly new light can be shed on this ongoing question.

While working cooperatively, humans predict movements of their fellow co-workers by experience and therefore find it easier to react. When robots work together with humans, human-like trajectories and dynamics are hence desirable. Especially humans that are well trained at certain tasks display extremely smooth and effortless motor control over a certainly very complex body, which is still far from being matched by any robot. Obtaining these smooth and natural movements has become a hot topic in robotics, especially when robots are supposed to work in close proximity to humans [128]. We can try to achieve this by actively computing human-like trajectories, but as long as the internal mechanisms of the robot are so different from the ones in the human body, this goal might never be truly achieved. A soft robot with human-like dynamics would enable the future service robot to work safely together with humans in the unstructured and dynamic environments of people's homes.

1.3 CONTRIBUTIONS

By drawing inspiration from biomechanical models and utilizing machine learning for obtaining relationships that are too complex to model analytically, an extensible control framework can be developed which offers guaranteed performance under certain assumptions. The control performance for musculoskeletal robots can be greatly improved by applying advanced techniques from the field of non-linear control. Due to the more complex structure, the feedback control laws from tendon-driven robotics deliver poor performance in musculoskeletal systems and can even become unstable, depending on the complexity of the robot. Even though it is well known that humans perform most of their fast movements, utilizing FFW control, it is clear that more fine grained move-

ments are only possible by the means of feedback control. In this thesis, state of the art feedback control laws for tendon-driven robots are compared with modern non-linear control techniques, showing great improvements in performance. The goal is to lay the foundation for controlling such robots which can be extended towards more complex tasks like impedance control or explosive movements. For this purpose a generic model for the class of musculoskeletal robots is developed, which can be used both for feedback as well as FFW control. It will become clear that the application of previously developed control laws to more complex musculoskeletal robots leads to instability of the closed loop dynamics, due to unmodeled effects. The novel control law, based on Dynamic Surface Control (DSC) which is an extension to *backstepping* solves these issues. It is further extended by introducing adaptive neural network control for the compensation of modeling errors due to friction.

The effects of colliding muscles, i. e. muscles wrapping around skeletal structures, are extremely difficult to capture in a analytical models. Hence the possibility of utilizing machine learning techniques is proposed, showing how it is possible to obtain reliable models of the muscle kinematics, i. e. the muscle lever arms. This even includes complex assemblies with spherical joints and multi-articular muscles. Results are presented for a musculoskeletal robot arm which has been developed specifically to capture the challenges present in this type of robots, namely complex joints and colliding muscles.

1.4 THESIS OUTLINE

This thesis comprises 9 chapters, starting after this introduction with Chapter 2 which presents an overview of tendon-driven and musculoskeletal robotics and related works in control, as well as the necessary background for machine learning that will be utilized in later chapters. In Chapter 3, the design of the musculoskeletal arm is presented which was used to obtain the results for the control techniques developed in this work. This includes the mechanics and sensory-motor system, as well as the electronic control architecture.

The dynamic system models are described in Chapter 4. These models are subdivided into several subsystems which can be analyzed separately, such as the skeletal dynamics, the muscle dynamics, and finally the muscle kinematics which can be obtained by different methods of machine learning.

In Chapter 5 the extensions for existing controllers based on the method of feedback linearization are presented. A novel control law is developed in Chapter 6 which utilizes the method of *backstepping* to obtain a fully non-linear controller. This controller is extended to account for the unmodeled effects of friction by introducing adaptive compensation terms in Chapter 7.

A comparison of the results for the different possibilities of controlling musculoskeletal robots developed in this thesis, is presented in Chapter 8. The performance of all feedback controllers is shown for the developed musculoskeletal robotic arm. Finally, concluding remarks are given in Chapter 9.

2

BACKGROUND

This chapter serves the purpose of giving some basic background over the techniques used in this work. First, an overview of existing tendon-driven and musculoskeletal robots is given in Section 2.1, presenting a categorization of the different kinds of robots. Subsequently the related work in tendon-driven robot control is discussed in Section 2.2, together with the related techniques from biomechanics, as well as the underlying principle of feedback linearization. Finally, an important factor in modeling musculoskeletal robots is the representation of the muscle kinematics, i. e. a geometric description of the tendon routing. This is especially difficult, as the utilization of complex joint types irrevocably leads to collisions between the muscles and the skeleton. These so called ill-defined muscle kinematics are extremely complex to capture in analytical models. In this work, utilization of machine learning is proposed for obtaining useful models. Therefore, an overview of the utilized techniques from this field is given in Section 2.3.

2.1 ROBOTS

2.1.1 *Tendon-Driven Robots*

Research on tendon-driven robots dates back to the 1970s and beginning of 1980s [86]. Force transmission through tendons had been identified as a possibility to reduce the weight of moving parts, e. g. for high speed manipulation or in the design for robotic hands. Tendon-driven robots can be differentiated by the tendon configuration, which is the number of tendons per DoF in the joints. Three main configurations exist, N, $N+1$, and 2N (see Fig. 2.1), while the first can also be classified as a flexible joint from the control perspective, since the tendon merely adds compliance into the drive chain. The $N+1$ configuration allows for more flexible placement of the actuators and exhibits the minimum number of tendons and actuators needed to control N positional DoF [50]. This is due to the fact that tendons can only pull, not push. The 2N configuration therefore facilitates the utilization of the additional control inputs for different purposes which include co-contraction of muscles for storing energy or controlling the joint impedance. For the latter, the number of tendons to control the full impedance matrix is $m = n(n+3)/2$, which amounts to $m = 2$ for a revolute and $m = 9$ for a spherical joint with three DoF [66]. Therefore, and due to the introduction of multi-articular muscles, the additional configuration of 2N+ is introduced, denoting many more muscles per DoF than 2N.

In the case of robotic hands it has been the tight size constraints that have been driving the research on tendon-driven technology. As early as 1986, a dextrous hand had

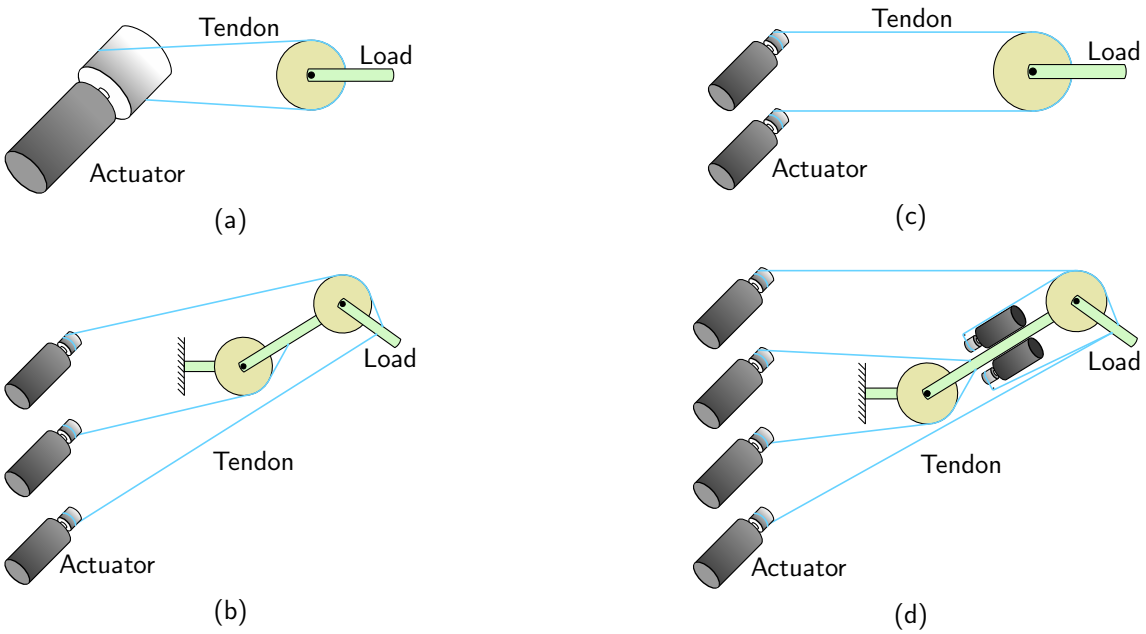


Fig. 2.1: Tendon Configurations. (a) Configuration N is a single actuator for N DoF. (b) Configuration N + 1 allows for tendon-driven control. (c) Configuration 2N is the minimum configuration to do impedance control. (d) Configuration 2N+ denotes many more actuators e. g. for multi-articular muscles or complex joint types. (adapted from [50]).

been built in collaboration between the Center for Engineering Design at the University of Utah, and the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology [49]. Tendon-driven technology enabled the placement of the motors outside of the hand, and therefore highly increased the number of possible DoF. For the Utah/MIT dextrous hand the DoF in the joints amounted to 19. Both the DLR, as well as NASA have been developing robotic hands to date, of which the DLR Hand II [74], the newer hand of the DLR Hand Arm System (HASy) [37], and the humanoid hand of the GM-NASA Robonaut-2 robot [1] are examples. Commercial products are available from the Shadow Robot Company which offers both pneumatic as well as electrical versions of their hands [117].

While robotic hands have become ever more complex over the years, larger tendon-driven robots have only recently come back into focus. In general tendon-driven robots can be categorized into several subcategories, depending on the tendon routing. Robots that have a design that is still relatively close to industrial manipulators have the N configuration, where the tendon is routed circular, leading from the actuator spindle to the joint spindle and back. The motor torque is directly transferred to the joint, depending on the radii of both the actuator and the joint spindle, allowing the motor to be placed apart from the joint. The most prominent example, here, is the BioRob, where tendons have been used to place the actuators into the base of the four DoF robot, reducing the total effective weight to only 0.5 kg for a load bearing capability of 2.0 kg [73]. Similarly, Isella 2

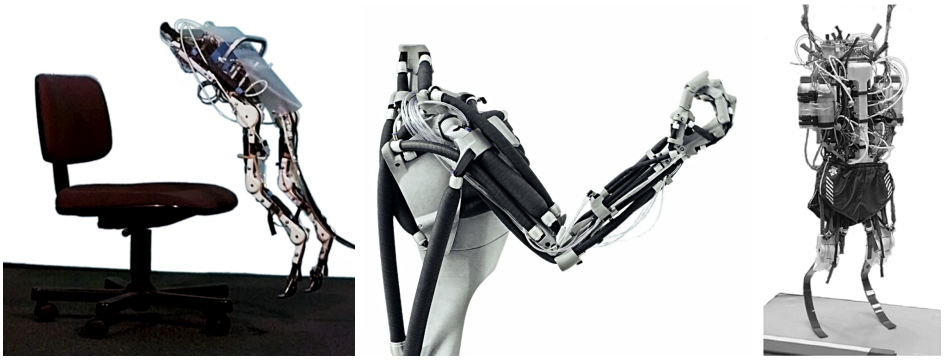
features tendon routing in a circular manner with a novel drive mechanism, the so called QuadHelix-Drive. Here, 4 tendons are coiled on a single actuator shaft, contracting two of them and extending the others. In Isella 2 these four tendons are used to drive a single joint, forming two circular tendon routes [111]. A more modular approach is available from IGUS, where an arbitrary number of 3d printed joints can be assembled. Each of the joints can have up to two DoF and is actuated by tendons that are guided by Bowden cable housings. Utilizing these modular joints, lightweight robots with a maximum number of five DoF can be built, where the motors are placed in the base of the robot. This configuration allows the assembly of robots that have a mass to payload ratio of 1 : 1 or even larger [47]. Utilizing Bowden cables, however, increases tendon friction significantly and therefore forbids applications, where (active or passive) compliance or force control is needed.

Utilizing the circular tendon routing, clearly extends the design space towards more lightweight robots. However, only biologically inspired antagonistic actuation yields the possibility to not only change the position of a joint, but also the compliance, as well as to store energy [6]. Here we can differentiate between a setup where the lever arm of the joint is constant over all joint angles, and where the lever arm is a function of the joint angle. Most of the hands, but also some tendon-driven manipulators include pulleys for routing the tendons, hence the lever arms are invariant. Generally control design is simplified by this choice, however, designing joints with multiple DoF is impossible without taking variable lever arms into account [66].

2.1.2 *Musculoskeletal Robots*

Musculoskeletal robots, feature the imitation of both a skeleton as well as muscles that are able to actuate it. In biological agents, the skeleton is made up of comparably stiff bones to which muscles are attached at insertion points or surfaces [36]. Flexing a muscle affects the torque of all joints that are located in between these insertion points, while muscles are called uni-, bi- or multi-articular depending on the number of joints that are crossed. Hence, musculoskeletal robots can be classified as those tendon-driven robots with the 2N+ configuration that also include multi-articular muscles.

The technology of musculoskeletal robots has been used to build biologically inspired robots in different fields of research. Especially the notion of Embodiment has driven the development of biologically and human inspired robots. A widely discussed line of thought in the field of Artificial Intelligence (AI) claims that it is not possible to recreate human-level artificial intelligence in a pure computational system, except for very limited and strictly confined fields like chess playing or expert systems. It is believed that intelligence needs a representation in the real world [15]. By mimicking biological systems we have learned to utilize the interaction of embodied agents with themselves and the environment to generate sensory feedback that truly represents the reality [103]. One



(a) (b) (c)

Fig. 2.2: Pneumatic Musculoskeletal Robots. (a) Mowgli, a bipedal jumping robot [93], (b) Airic, a musculoskeletal shoulder complex [26], and (c) a musculoskeletal athlete robot, designed for running [94] are shown.

goal is to understand how such complex bodies as our own, or of other biological beings, can be controlled by a brain.

Already relatively simple musculoskeletal robots like Mowgli (see Fig. 2.2a) have been shown to outperform classically engineered robots by exploiting the inherent dynamics of the mechanics. It was observed that the movements of jumping and landing demand large forces in the muscles for very short periods and the energy expenditure of repetitive jumping and landing cycles can be decreased by storing energy in between cycles. The musculoskeletal structure consists of six McKibben pneumatic actuators including four bi-articular muscles, allowing for pre-tensioning before take-off. The structure was shown to efficiently decrease the control effort for the performed movements, as there was no need to directly control the timing of take-off and landing. Instead, this effort was off loaded into the morphology of the structure [93].

In robots like Mowgli bi-articular muscles play an important role. It has been shown by Gregoire et al. [38] that mono-articular muscles in the human knee and the ankle cannot deliver much force and cannot explain the power output in jumping movements. Furthermore, multi-articular muscles can serve in the synchronization between joints in explosive movements [134], as well as in reaching tasks [62]. It is widely believed that bi-articular muscles deal with large scale movements of multiple joints. Several groups have replicated parts of the human body utilizing McKibben actuators to investigate these hypotheses. The automation technology supplier Festo built the Airic (see Fig. 2.2b), a full shoulder complex with 30 muscles (including bi-articular) in 2007 [26]. A musculoskeletal athlete robot was built by Niiyama et al. [94], investigating explosive movements in human running (see Fig. 2.2c). After starting from a catapult to provide a stable start, it was possible to demonstrate the robot running for five steps by pure FFW control. Similarly, Hosoda et al. [46] built an anthropomorphic musculoskeletal upper limp. The main goal was to understand embodied intelligence, by investigating soft interaction with the

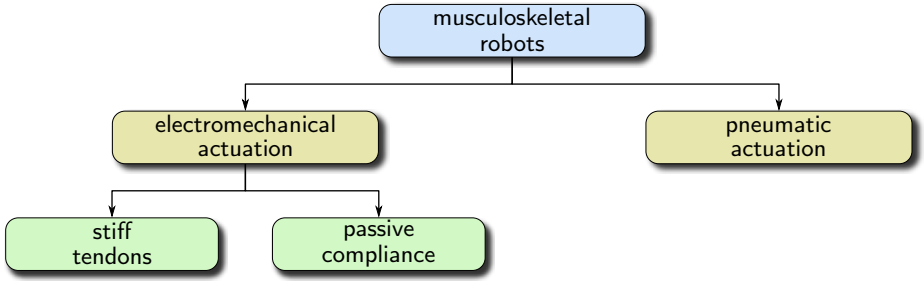


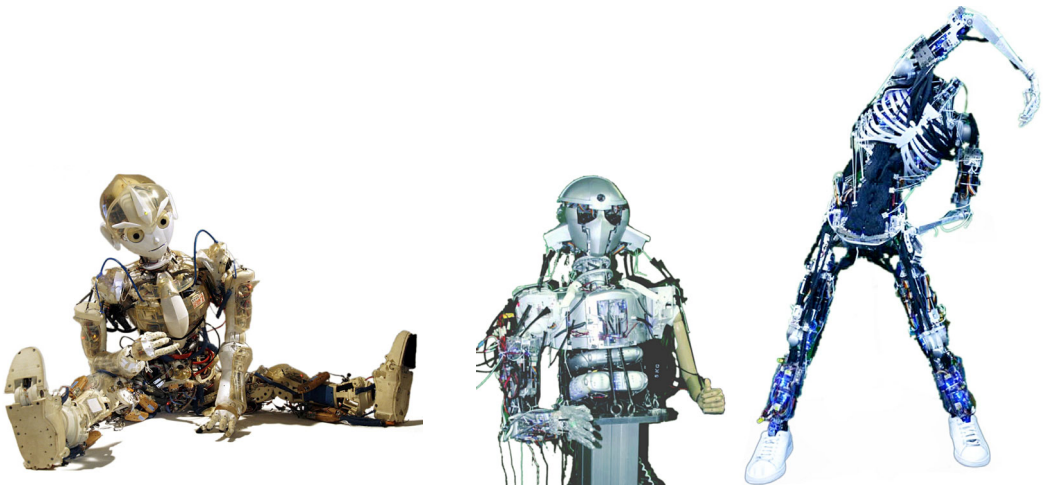
Fig. 2.3: Musculoskeletal Robots can be categorized by the type of actuation, as well as the tendon compliance.

environment. The humanoid robot arm consists of a shoulder, an upper arm, an elbow, a forearm, a wrist and a hand. Excluding the hand, the arm has 7 DoF, with a total of 14 muscles.

Pneumatic artificial muscles have been developed to closely replicate the mechanical properties of biological muscles. A rubber inner layer, wrapped in a fiber shell contracts in length and expands radially when air pressure is applied. The produced force is both subject to the extension and the applied pressure. The main drawback, however, is the controllability of such a muscle which is diminished by the highly non-linear input-output relationship. This relationship changes with the age of the rubber, as well as with the temperature. Furthermore, valves are used to control the pressure that can only be opened and closed at frequencies of approximately 40 Hz to 50 Hz [16]. Even though, advanced behavior could be demonstrated with musculoskeletal robots with pneumatic actuators, the movements are mostly FFW and cannot be controlled in a feedback manner.

Unlike pneumatic actuators, musculoskeletal robots with electromechanical drives consist of an electromagnetic motor that winds a cable on a spindle and hence either innervates or relaxes the artificial muscle, depending on the direction of motor rotation. Radkah et al. [107] have built BioBiped, a fully compliant, tendon-driven robot to investigate robot walking with this type of actuator. Especially in walking, the energy expenditure as well as overall performance of robots is still far below human level. Similar to the musculoskeletal athlete robot, BioBiped proves that utilizing compliant tendon-driven technology with bi-articular muscles is the key towards human-like performance in locomotion. An intelligent mixture of active and passive actuation both uni- and bi-articular, led to a stable hopping behavior. The actuators used in this robot were categorized as bi-directional Series Elastic Actuator (SEA), uni-directional SEA, mono-articular passive structure, and bi-articular passive structure. While the first falls into the category of circular tendon configuration (N configuration), the latter three utilize the 2N+ configuration with variable lever arms.

The Jouhou System Kougaku Laboratory of the University of Tokyo (JSK) has developed several generations of musculoskeletal robots with electromechanical drives. Starting with Kenta in 2002 [83], a robot with a flexible spine and a total of 96 muscles was



(a) (b) (c)

Fig. 2.4: Musculoskeletal Robots at the JSK. The robots (a) Kotaro (2006) [84], (b) Kenzoh (2011) [89] and (c) Kenshiro (2012) [67] are shown.

built. Development continued with Kotaro [84] for the World Exhibition (EXPO) in 2005 (see Fig. 2.4a). Improvements included predominantly the mechanical structure which resulted for the first time in a floating shoulder blade. Most parts of the skeleton were 3d printed utilizing Selective Laser Sintering (SLS). Kojiro appeared only one year later with improved actuation due to the application of brushless Direct Current (DC) motors instead of the standard brushed DC motors, previously used [85].

Kenta, Kotaro and Kojiro were musculoskeletal robots, but they did not include compliant actuation, by introducing elastic materials. Therefore they belong to the category of musculoskeletal robots with stiff tendons (see Fig. 2.3). In 2011, Kenzoh was developed, reflecting both the human power output as well as the compliance. The goal was to investigate dynamic tasks where pre-loading of muscles, impedance control, as well as the passive shock absorption are of interest. For the latter, decoupling between actuator and joint by an elastic element is needed to protect the gearbox from high peaks in cases of collisions. The same is true for the storing of energy, while it is fully sufficient here to include linear springs. However, when impedance control is considered, co-contraction of muscles has to lead to stiffer joint behavior. This is not the case when linear springs are added. For a simple example of a revolute joint with point to point routing and fixed lever arms (see Fig. 2.5), the torque τ can be expressed as

$$\tau = r_j \left[k_1(\theta_1 r_s - q r_j) - k_2(\theta_2 r_s + q r_j) \right] \tag{2.1}$$

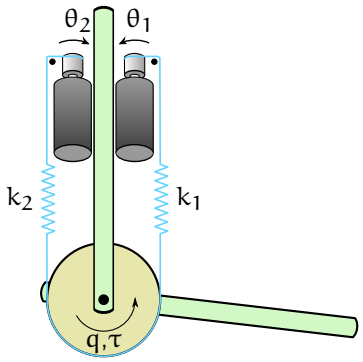


Fig. 2.5: Antagonistic actuation. By expressing the stiffness of the joint with respect to the actuator positions θ_1, θ_2, it can be seen that the joint stiffness cannot be controlled directly when linear spring stiffnesses k_1, k_2 are assumed.

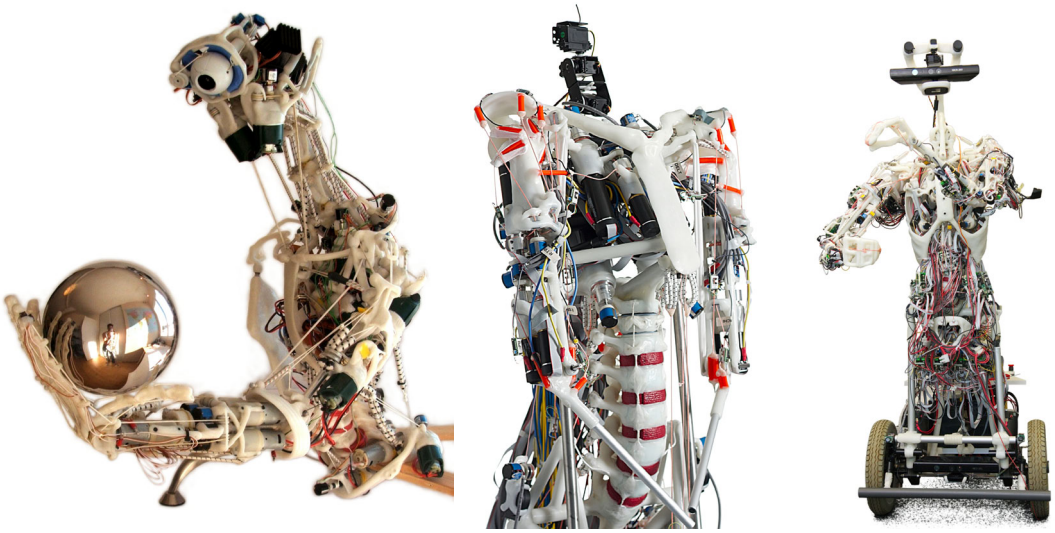
with r_j as the fixed muscle lever arm and r_s as the actuator spindle radius. The joint stiffness k_q is then defined as the partial derivative of the torque with respect to the joint angle.

$$k_q = \frac{\partial \tau}{\partial q} = -r_j^2(k_1 + k_2) \tag{2.2}$$

The negative sign arises from the definition of the torque in the same direction as the joint angle, i. e. the stiffness counteracts a positive joint deflection with a negative torque. From Eq. 2.2 it is observed that if the spring constants of the muscles are constant, i. e. springs are linear, joint stiffness is also constant. Applying this to robots with variable lever arms, the joint stiffness will not be constant, as lever arms change with the joint angle. However, joint stiffness remains uncontrollable as actuator positions do not directly affect the lever arms. Only if the spring constants depend on the expansion, the joint stiffness becomes a function of the actuator positions θ_1, θ_2 and is therefore adaptable by co-contraction. Usually progressive springs are chosen to achieve low stiffness at low co-contraction, examples of which are [63, 80].

In 2012 the newest musculoskeletal full body robot of the JSK, Kenshiro came into existence. The aim of developing a humanoid that can be used as a human body simulator led to a robot with bi-articular muscles and movement capabilities that are very close to its human role model. To achieve this, planar muscles were introduced which are capable of acting not only upon a single attachment point, but on a line, by routing the cable back and forth between the attachment areas. This also has the advantage of introducing an additional reduction ratio, improving overall efficiency [11, 67]. However, the notion of elastic elements in the muscles was dropped.

In 2006 Holland and Knight [43] came up with the *Anthropomimetic Principle*. The initial goal was to create a robot that was capable of developing self consciousness. Since the only creature that was proven to be self conscious is the human, a robot torso was developed that was as close to the human body as possible. This meant not only copying the



(a) (b) (c)

Fig. 2.6: ECCE Family. (a) The first anthropomimetic robot CRONOS [43] (b), the ECCE-3 and (c) the ECCE-4 is shown.

outer envelope of the human as it is done by most other humanoid robots, but also mimicking the inner structures, like bones, muscles, and tendons. The first anthropomimetic robot, named Conscious Robot (Or Near Offer) System (CRONOS), consisted of a hand modeled skeleton and muscles that consisted of cheap screw driver motors, kite line and shock cord as an elastic element. The thermoplastic material, used for the skeleton can be molded by hand at a temperature of 60°. For the upper torso a number of 45 muscles was replicated. In contrast to the JSK robots at that time, where the focus was on the functionality, the main concern for the development of CRONOS was to replicate the skeleton and muscle insertion points, and therefore achieving dynamics that were very similar to the human body. The muscles that were chosen to be duplicated in this robot were the ones responsible for larger scale movements, omitting the ones used for fine grained dexterous movements, e. g. in the hands. In all robots, as well as humans, proprioception—the sense of the relative position of neighboring parts of the body—is fundamental for well-controlled movements and interactions with the environment. While CRONOS was only sensorized by potentiometers on the gearbox shafts, to measure the actuator positions, the project was carried on within the Embodied Cognition in a Compliantly Engineered Robot (Eccerobot) framework, where sensorization, control, and simulation of robots just like CRONOS was investigated [76]. Several robot torsos (see Fig. 2.6) were developed that were subsequently equipped with ever better sensors and actuators, starting from CRONOS which was retro-fitted with additional sensors and hence became the first generation of the ECCEs. While the second generation, the Eccerobot Design Study (EDS) (see Fig. 1.1c) was mainly meant to improve the mechanical structure, in the ECCE-3 (see

Fig. 2.6b), the screw driver motors had been replaced by high-quality DC motors with shaft encoders. It featured a full-fledged spine with a total of 16 actuators. Altogether the robot was actuated by 48 muscles. The next step was to equip the next robot—the ECCE-4 (see Fig. 2.6c)—with force sensors in all muscles. Next to the spine, a floating shoulder blade was introduced to cover the full range of human arm movement [143] increasing the number of muscles to 52. All robots of the ECCE family can therefore be categorized as musculoskeletal robots with electromechanical drives and passive compliance.

Within the Eccerobot project the use of different flexible materials for adding the compliance to the muscles have been investigated. The goal was to improve the efficiency and safety of the robots, as well as increasing the possibilities of adapting to uncertain environments. Starting from shock cord as it is used in marine applications which has been shown to behavior linearly within 80 % of the resting length [140], the need for a progressive spring relation brought forward a Polyurethane used for conveyor belts. However, the latter showed highly unwanted plastic deformation which made obtaining a model difficult. Furthermore, the inexistence of Computer Aided Design (CAD) data made (1) obtaining kinematic and dynamic models an extremely time consuming and error prone task [140] and (2) diminished the producibility of the robots.

2.2 CONTROL

The topic of control for tendon-driven robots has been ongoing work in research for several decades. While benefits were seen mainly in being able to actuate manipulators remotely [50] and therefore reducing the active inertia of the robots, the elasticity of tendons was identified as a major drawback, together with the problem of slackening tendons. With the realization of musculoskeletal robots, the elasticity of the tendons is a wanted feature that first needs to be handled and later exploited by control algorithms.

In the following, a short introduction into feedback linearization is given which was used by most control algorithms for tendon-driven robots to date. Subsequently, a summary of the related works from the field of tendon-driven control is given in Section 2.2.2. Similar techniques have been utilized by biomechanicists on simulated human bodies for the purpose of gaining physically and physiologically realistic movement data. These applications show the scalability of this approach and are hence presented in Section 2.2.3. Finally, possible utilization of the existing techniques for controlling musculoskeletal robots is discussed in Section 2.2.4.

2.2.1 *Feedback Linearization*

Possibly the simplest method to find a control law for a system that exhibits non-linear dynamics, is the method of feedback linearization. Here, the known non-linear terms are compensated by the controller, to subsequently apply linear methods to stabilize the linearized system. It requires the system to exhibit a certain structure. However, there is a

set of well-defined tools to apply coordinate transforms to a wider range of systems such that the method can still be applied [60]. The necessary structure is as follows,

$$\dot{x} = Ax + B\gamma(x)[u - \alpha(x)] \tag{2.3}$$

where x is the system state and u the control input. If this is the case, a state feedback law can compensate for the non-linear terms α and γ by subtraction and division, respectively,

$$u = \alpha(x) + \gamma^{-1}(x)v \tag{2.4}$$

where v is now the new control input that can be used to stabilize the linearized system.

In robotics a wide range of controllers have been developed of which most can be seen as special cases of computed torque control which is an application of the method of feedback linearization [119]. The equation of motion in the canonical form for a torque controlled robot can be given as follows [119],

$$\tau = H(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_G(q) \tag{2.5}$$

while $H(q)$ denotes the mass matrix, $C(q, \dot{q})$ accounts for Coriolis and centrifugal effects, and $\tau_G(q)$ is the vector of gravity terms. Eq. 2.5 can be rewritten to exhibit the structure necessary to apply feedback linearization.

$$\begin{bmatrix} \ddot{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ I_n & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{q} \\ q \end{bmatrix} + \begin{bmatrix} H^{-1}(q) \\ 0 \end{bmatrix} [\tau - C(q, \dot{q})\dot{q} - \tau_G(q)] \tag{2.6}$$

Where I_n denotes the $n \times n$ identity matrix. The control law found by this method computes a torque τ which is necessary to achieve a reference joint acceleration $v = \ddot{q}_{ref}$ (see Fig. 2.7).

$$\tau = H(q)v + C(q, \dot{q}) + \tau_G(q) \tag{2.7}$$

Hence, the now linearized system can be stabilized with e. g. a PD controller, utilizing the virtual control input v. Convergence can be proven, utilizing linear system theory. However, perfect system models are crucial to fully compensate the non-linear dynamics. In case of deviations in the model parameters or unmodeled dynamics (e. g. friction) the assumptions for the proof of stability do not hold and the closed loop dynamics might become unstable.

2.2.2 *Tendon-Driven Robot Control*

Jacobsen et al. [50] developed a linear controller for a single joint with two tendons, showing a performance that was considered to be as good as a system with a single joint actuator. However, neither gravity effects nor coupling between multiple joints was taken

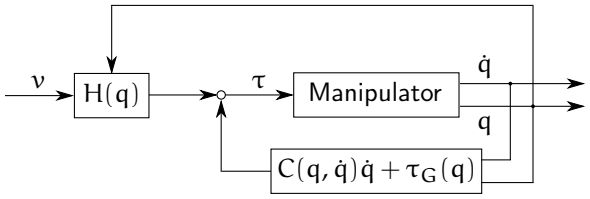


Fig. 2.7: Computed Torque Control (adapted from [119]). The dynamic model of the manipulator in the canonical form is used to compute a torque τ that is needed for a reference acceleration ν. Hence, the Coriolis and centrifugal terms $C(q, \dot{q})$ and the gravity terms $\tau_G(q)$ are used together with the mass matrix $H(q)$ to linearize the system.

into account. Similarly, Potkonjak et al. synthesize a robust controller for an isolated joint, using the so called puller-follower concept, where one of the muscles—the puller—is exerting a certain force and the follower keeps the muscle from going slack [105]. This approach was extended to multi-joint systems, by introducing gravity compensation [106], without considering more complex joints or multi-articular muscles.

In contrast, controllers for tendon-driven hands have been developed invariably by utilization of feedback linearization for the joint dynamics (see Section 2.2.1) and low-level tendon control. The joint dynamics can be simplified for this case, since the joint torques due to Coriolis and centrifugal components are negligible compared to the gripping forces. Salisbury and Craig [113] were the first to implement a tendon-level control scheme, using the Stanford/JPL hand which was an $N+1$ configuration. Since the joint dynamics controller, based on feedback linearization, utilizes the joint torque as control variable, a matrix of the constant lever arms M was introduced to capture the effects of the tendon routing, such that

$$M \cdot f = H(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_G(q) \tag{2.8}$$

where f is the vector of tendon forces. Here, M is in $\mathbb{R}^{n \times m}$, with n as the number of skeletal DoF and m as the number of tendons. The underspecified problem of computing tendon tensions for joint torques was solved by setting one of the tendon tensions to a positive value. For $N+1$ configurations this ensures positive tendon tensions. Subsequently the inverse of the remaining tendon lever arm matrix was used to compute the other tendon tensions. Each tendon was equipped with an independent PI controller with an additional FFW term. Lee et al. [72] propose a similar controller which features additional stiffness control for a 2N tendon-driven robotic hand. The problem of inverting the matrix of tendon lever arms is solved by utilizing the weighted pseudo inverse and an offset for the constraint of positive tendon tensions. The controller includes a position estimation algorithm for approximating the joint angles from the measured motor positions and the tendon tensions. The low-level tendon tension controller was implemented as a PID control law with FFW part. Based on this work, Abdallah et al. [1] proposed a new type of controller, where the low-level tension control is replaced by an actuator position controller. Here, the reference actuator positions were computed from

the tendon stiffness and the reference tendon force. The problem of inverting the matrix of lever arms was solved utilizing the pseudo inverse for the $N + 1$ configuration. It was extended in [2] by a sophisticated optimization procedure which can be used in real time to compute minimum tendon forces for the reference joint torques.

Closely related to the control problem for tendon-driven hands is the one for parallel cable-driven robots which are used for high-speed manipulation [58]. These manipulators are usually implemented with an $N + 1$ configuration to minimize the number of actuators, while the weight of the moving parts is reduced by transferring the actuators to the static part. Fang et al. [23] proposed a trajectory tracking controller with optimal tension distribution, utilizing a pose-dependent matrix of lever arms. The full manipulator dynamics were used for feedback linearization, in conjunction with a low-level tendon force controller.

The feedback linearization schemes presented so far assume perfect low-level tendon control, since the design of the two control layers is conducted independently. In contrast, Palli et al. [99] proposed a feedback linearization controller for the full system dynamics, including the actuator dynamics. In this implementation, the control law was formulated for an arbitrary assembly of revolute joints with two antagonistic actuators, each, even though extensions for more complex assemblies could possibly be developed.

For the control of a tendon-driven mechanism with two revolute joints and six muscles (including 2 bi-articular muscles) Kobayashi et al. showed an adaptive control scheme [63]. Here, the joint, as well as the actuator dynamics were approximated by Radial Basis Function Networks (RBFNs) (see Section 2.3.3), utilizing the trajectory tracking errors as inputs for the respective gradient descent learning rules. Hence, the plant dynamics could be learned during controller operation. The passivity based control law utilized the pseudo inverse of the matrix of lever arms and an offset for positive tendon forces. It has to be noted that the matrix of lever arms was constant in the joint angles and not part of the adaptation process.

Mitrovic et al. [81] proposed a supervised learning approach to acquire the plant dynamics as well as its stochastic properties, for a planar 1 DoF joint with two antagonistic actuators. The system considered here was relatively simple, featuring a 2N approach and only one revolute joint. Furthermore, the controller did not have to deal with gravity effects, due to the planar nature of the setup. However, the system dynamics were acquired completely by machine learning techniques, including the muscle kinematics. This learning approach was extended for the simulation of a tendon-driven robotic arm with two DoF and six muscles, two of which were bi-articular [82]. It could be shown that it is possible to simultaneously learn the forward dynamics of a tendon-driven system and apply Optimal Feedback Control (OFC). However, the curse of dimensionality makes real-time execution of such a control scheme for more complex robots impossible at this stage.

2.2.3 *Biomechanics*

Biomechanicists utilize methods, that are quite similar to the tendon-driven controllers presented in the previous section, for extracting muscle excitation patterns for given joint space or task space trajectories. It could be shown that it was possible to obtain physically and physiologically realistic data by running robotics control algorithms on simulations of the human body. Even though the control task does not need to be robust in any way, since models are perfectly known in simulation and no disturbances are introduced, the complexity of the utilized simulation models exceed by far the tendon-driven robots developed up to date.

Even though developed for comparably simple tendon-driven robots, biomechanicists extended the approach of utilizing a high-level feedback linearization controller and low-level tendon controllers for more complex simulation scenarios. Hence, it could be shown that, at least in simulation, these algorithms scale very well with the number of joints and muscles. Furthermore, complex joint types like the human shoulder joint, multi-articular muscles and highly non-linear tendon routing schemes were introduced. Even though these are exactly the challenges of the control problem of musculoskeletal robots, one has to bear in mind that the results contain no evidence of the robustness, necessary for physical implementations. This is due to the fact that perfect system models are always available in simulation.

In Thelen et al. [130, 131], a method called *computed muscle control* was developed to compute muscle activations for given kinematics from human motion data. This well-known problem, when dealing with simulations of the human musculoskeletal system, had previously been solved using dynamic optimization methods, which are extremely computationally expensive. The presented method utilized a low-level muscle activation controller and a feedback linearization controller for the high-level skeletal dynamics, similar to the previously presented controllers for tendon-driven hands. However, the complex relationship between joint torques and muscle forces was captured by the so called muscle Jacobian (see Section 4.3 and [149]), which can be computed from the geometric properties of the muscle routing, even for muscles that collide with the skeleton. The under-determined problem of computing muscle forces for joint torques was solved utilizing quadratic optimization. This approach was extended by de Sapio et al. [21, 22] for controlling the simulation of a human shoulder-arm complex. It could be shown that this approach in combination with the optimization criterion of minimizing the normalized muscle activation leads to good correlation between human reaching postures and the resulting postures of the simulated shoulder model [22]. Furthermore, operational space control for the human shoulder-arm complex could be shown [21]. Especially solving the problem of muscle as well as joint redundancies in a single optimization step is of interest for the control of musculoskeletal robots.

Aside from Thelen, there were other biomechanicists who explained the movement generation of the human body by controlling a simulated musculoskeletal human model consisting of a *musculo-tendon network* and a skeleton. Yamane and Nakamura [146] took

control algorithms from robotics to control joint torques and a matrix of lever arms obtained from the simulated model to realize physically and physiologically realistic movements for a full human model with a total of 997 muscles. Here the approach was to optimize the muscle tensions towards measured Electromyography (EMG) data for the muscle activations that were measurable and explain the remaining activations with the physically consistent model.

In [128], Tahara et al. proposed a task space control scheme for a musculoskeletal planar arm, that overcomes the issues of joint and muscle redundancies. It was shown that the *virtual spring-damper hypothesis* [9] in combination with bi-articular muscles leads to the slightly curved trajectory of human reaching movements, without the need for complex optimization. The shape of the trajectory could be altered by the magnitude of the internal forces and therefore the stiffness of the arm. Later work based on this, namely Kino et al. [62], showed how the introduction of bi-articular muscles improves controllability of a musculoskeletal arm, utilizing the Equilibrium Point Hypothesis (EPH) [25] and Tahara et al. [126, 127] focused on learning stiffness ellipsoids for given tasks. In contrast to the other biomechanical work, presented so far, Tahara et al. verified the control approach for the physical implementation of a 2-link planar arm with six muscles (including 2 bi-articular muscles) [129].

2.2.4 *Discussion*

Looking at the musculoskeletal robots like the ECCEs or the family of robots built at the JSK (see Fig. 1.1c and Fig. 1.1b, respectively), it becomes obvious that controllers developed for such structures need to solve the identical control challenges, also present in humans. There have been several hypotheses on how humans are able to control their very complex bodies, namely the EPH [25] and the Internal Dynamics Model Hypothesis (IDMH) [59], however up to date the details of human motor control remain unknown. It is clear that the control problems for spherical joints, multi-articular muscles, and variable lever arms have to be solved in a robust way, even for ill-defined muscle kinematics, i. e. muscle kinematics that are difficult to model due to collisions of the muscles with the skeleton. The control approaches presented in Section 2.2.2 show several feasible approaches, like the separation into high-level and distributed low-level control or the adaptation to unknown parameters. However, only the techniques from biomechanics exhibit the necessary flexibility and scalability due to the utilization of the muscle Jacobian. Table 2.1 gives an overview over the robots and biomechanical models covered so far, in comparison with the class of musculoskeletal robots.

The work on control for actual musculoskeletal robots is at a very preliminary stage, e. g. presenting the interpolation of actuator positions between previously gathered samples and subsequent FFW control [75]. Similarly, all the presented applications for musculoskeletal robots have been implemented in a FFW manner [46, 88, 92, 107], failing by far to reach human-level control performance. It is obvious that when reaching tasks are

Table 2.1: Robot/Simulation Model Configurations. The table shows the complexity of the controlled tendon-driven robots in comparison to the presented biomechanical models and the musculoskeletal robots for which a controller is to be found. The 2N+ configuration denotes systems where the number of tendons is larger than twice the number of joint DoF.

	conf.	joints	lever arms	multi-art. muscles	colliding muscles
Jacobsen et al. [50]	2N	revolute	constant	○	○
Potkonjak et al. [105, 106]	2N	revolute	constant	○	○
Salisbury & Craig [113]	N + 1	revolute	constant	○	○
Lee et al. [72]	2N	revolute	constant	○	○
Abdallah et al. [1, 2]	N + 1	revolute	constant	○	○
Fang et al. [23]	N + 1	6-DoF	variable	○	○
Palli et al. [99]	2N	revolute	constant	○	○
Kobayashi et al. [63]	2N	revolute	constant	●	○
Mitrovic et al. [81, 82]	2N+	revolute	variable	●	○
Thelen et al. [130, 131]	2N+	spherical	variable	●	●
de Sapio et al. [21, 22]	2N+	spherical	variable	●	●
Yamane & Nakamura [146]	2N+	spherical	variable	●	●
Tahara/Kino et al. [62, 126–128]	2N+	revolute	variable	●	○
Musculoskeletal robots [67, 143]	2N+	spherical	variable	●	●

considered, the sole use of FFW control is insufficient, even though it is well understood that large parts of human control utilizes FFW models, i. e. the utilization of a learned inverse dynamics model to compute motor commands for planned movements. Robust and scalable feedback control laws that utilize the dynamics in a FFW manner for musculoskeletal systems are lacking up to now, especially when complex joints and ill-defined muscle kinematics are considered.

2.3 MACHINE LEARNING

Up to now most robots have had stiff kinematics, where motion kinematics and dynamics can be established analytically, and even for the available soft robots like the DLR light weight robot arm or variable stiffness actuators, the flexible joints can still be modeled in a straightforward manner [7]. However, there are also problem statements where an algorithmic solution is not feasible, as models become too complex to be properly parametrized. In these cases, machine learning techniques can help, provided that enough data can be generated for the learning algorithms to work with. Even though the data might not account for all effects in the future, machine learning algorithms are well able to construct a good and useful approximation of the underlying unknown processes [8].

In general, machine learning algorithms can be classified as one of the following: (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning. In supervised learning, the goal is to optimize the output for given inputs towards known correct values that are given by a supervisor. This is not true for unsupervised learning. Here, the algorithm has to find structures in the input space on its own, by identifying certain reoccurring patterns. The last category—reinforcement learning—can be applied to problems where a goal can be reached by a sequence of actions. The algorithm solely requires a measure of how good or bad an ordered set of actions performed, either during execution or at the end of the task. For instance these actions can be moves in a game, which lead to winning or losing the game. Apart from games like chess, reinforcement learning has lately been applied to tasks in robotics, where a goal, e. g. a reaching task, is defined and the algorithm has to find a sequence of motor commands to perform the task without knowing anything about the system [61, 101]. While the latter approach is promising for future application, the main problem is that a large number of trials need to be performed to learn a single task. Safe system operation can in general only be guaranteed in simulation. Furthermore, the search space needs to be simplified by introducing so called motor primitives [65]. In the case of musculoskeletal robots, these motor primitives have to be established first, before an application of these techniques is even possible.

In this work, however, the problem statement is to learn a certain part of the kinematic model—the muscle kinematics—which cannot be modeled by conventional analytic methods. This can be achieved by gathering samples from the robotic system, both for the system inputs, as well as the system outputs. Therefore, the problem at hand, is a supervised learning problem. In supervised learning there are two separate problems, (1) classification and (2) regression. In the first, samples are considered to either be in a given class or not, the output is boolean. If the output is numerical, it is called a regression problem, or in other words the goal is to approximate a numerical function which is able to predict the mapping between in the input and the output space. The training set can be defined as follows,

$$X = \{x_t, r_t\}_{t=1}^{n} \tag{2.9}$$

where x_t are the function inputs and r_t the corresponding outputs. If samples could be drawn without the effects of noise, the problem could be stated as an interpolation problem where an $n-1$ degree polynomial can be used. It can be fitted by solving a set of linear equations of order $n-1$. However, in real world problems there is always measurement noise and this procedure would lead to an overfitting of the function, i. e. learning the function including the errors due to the noise. Therefore, we draw a lot more samples and use methods from regression to fit a function

$$y = f(x|c) \tag{2.10}$$

where c are the parameters for the model $f(\cdot)$. Here, $f(\cdot)$ is the regression function, for which the machine learning algorithm optimizes the parameters c towards an optimal

value of a predefined error measure. A common error measure is the Mean Square Error (MSE).

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} \|f(x_t) - r_t\|^2 \tag{2.11}$$

In some cases it makes sense to write the error in units that match the data, hence utilizing the Root Mean Square Error (RMSE).

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{2.12}$$

In the following, three methods for regression are covered, starting with the simple method of polynomial fitting (see Section 2.3.1), subsequently discussing Feed-Forward Neural Networks (FFW-NNs) in Section 2.3.2 and finally RBFNs are introduced in Section 2.3.3.

2.3.1 *Polynomial Regression*

A simple solution for the regression problem for scalar inputs and outputs is polynomial function fitting, where a predefined polynomial of order m is fitted.

$$g(x, a) = \sum_{j=0}^{m} a_j x^j \tag{2.13}$$

Fitting can be achieved by minimizing an error function which is a measure of how good the fitted function performs for the given samples. A common choice is the MSE (see Eq. 2.11) which can be rewritten for the scalar case.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} (g(x_t, a) - y_t)^2 \tag{2.14}$$

The problem of polynomial fitting can be solved by choosing a vector a that minimizes the error function. Due to the fact that this error function is quadratic in the coefficients a, its partial derivative $\frac{\partial \text{MSE}}{\partial a}$ is linear. Therefore, the optimization problem has a unique solution [13]. Using matrix formulation, a linear equation can be set up

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix}}_{X} \cdot \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}}_{a} \tag{2.15}$$

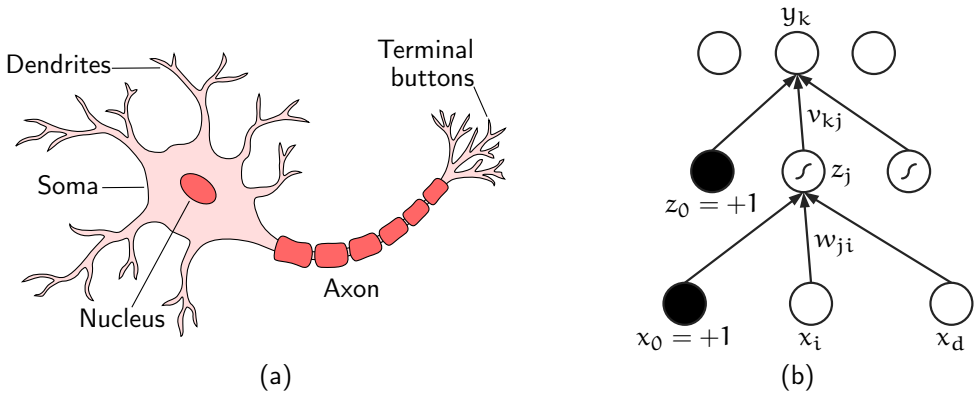


Fig. 2.8: Neural Network. A (a) biological neuron (adapted from [56]) is shown next to (b) a graphical representation of a Feed-Forward Neural Network (adapted from [8]).

that is overdetermined for regression problems ($n > m + 1$); The number of samples is usually a lot larger than the degree of the polynomial plus one. Therefore, an approximation for a can be found, using the generalized inverse $\overline{X}$.

$$\hat{a} = \overline{X} \cdot y = (X^T X)^{-1} X \cdot y \tag{2.16}$$

Note that solving the linear problem $y = X \cdot a$ through the generalized inverse is equivalent to minimizing the MSE [69].

The method of polynomial fitting can be used for scalar functions, where all samples are gathered, and the regression problem is solved once, as is the case in any *offline* learning problem. If necessary this method can be extended to the *online* case by utilizing a Recursive Least Squares (RLS) algorithm [55].

2.3.2 *Feed-Forward Neural Networks*

Artificial Neural Networks (ANNs) take inspiration from the human brain in which a very large number (10^{11}) of neurons operate in parallel [8]. These neurons are interconnected by *synapses* which are believed to be both responsible for the computational power, as well as the memory. Interconnections can grow dynamically by impinging terminal buttons upon another neuron's dendrites or soma (see Fig. 2.8a and [56]). The goal of ANNs in general is not to simulate the functionality of the human brain, but to mimic some aspects in order to develop algorithms that can learn complex relations similar to humans. While different kinds of ANNs have been developed to cover different aspects of the functionality in biological neural networks, the most simple, and most widely used kind, is the FFW-NN. In this type of neural network, neurons are solely interconnected from one layer to the next, without forming feedback connections (see Fig. 2.8b). The reason is, that these networks are simple to evaluate, and more importantly, still relatively simple to train and can be used to recognize complex relationships in a set of samples or to classify

patterns. In the following, a special type of FFW-NN, the Multilayer Perceptron (MLP), is introduced.

2.3.2.1 *The Multilayer Perceptron*

An MLP has a set of input neurons—forming the input layer—which are used to present the input data to the network. Each neuron is connected to each neuron in the subsequent layer by directed weighted connections. While there can be an arbitrary number of so called hidden layers, the last layer is called the output layer. In the simplest case, the output of a neuron o_j is only the sum of the inputs, while an additional neuron with output $+1$ is added for each layer, to be able to introduce an offset.

$$o_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0} \tag{2.17}$$

Where w_{ji} denotes the corresponding layer weight, x_i the activation of the i^{th} input layer neuron and d the dimensionality of the input layer (see Fig. 2.8b). More general, a neuron can be associated with an activation function, which describes the relationship between the output of a neuron and the sum of the inputs. Assuming a linear activation function, as in Eq. 2.17 leaves the function of the neural network to be linear, as well. For the purpose of modeling non-linear functions it is therefore necessary to find suitable non-linear activation functions. The activation a_j of a neuron j is given as follows, defining the input to the activation function f_a.

$$a_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0} \tag{2.18}$$

$$o_j = f_a(a_j) \tag{2.19}$$

Possible functions include the threshold or the piecewise linear function, but also the tangens hyperbolicus, or the logistic sigmoid function. All of these have the property to squeeze an infinite input range into a finite output range which is a useful property for activation functions. As we will see later in Section 2.3.2.2, it is important that the activation functions are differentiable, as training algorithms rely on the gradient to adjust the weighted connections towards a given goal. For this reason, mostly the logistic sigmoid function or the tangens hyperbolicus are used. While the output of the former ranges from 0 to $+1$, the latter ranges from -1 to $+1$. The sigmoid function σ and the tangens hyperbolicus tanh and their respective derivatives are given as follows.

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x)) \tag{2.20}$$

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \qquad \frac{d}{dx}\tanh(x) = 1-\tanh(x)^2 \tag{2.21}$$

It can be proven, that a single hidden layer is sufficient to approximate any non-linear function, given the hidden layer contains enough neurons [45]. Therefore MLPs are a class of universal approximators. In the case of a single hidden layer, the outputs of the hidden layer neurons z_i can therefore be expressed with respect to the input neurons x_j and the corresponding weights w_{ji}.

$$z_j = f_a\left(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right) \tag{2.22}$$

Subsequently this enables the computation of the output layer y_j, subject to the hidden layer neurons z_k and the corresponding weights v_{jk}. Note that in this case it is assumed that the output layer has no activation function (see Fig. 2.8b).

$$y_k = \sum_{j=1}^{d} v_{kj}z_j + v_{k0} \tag{2.23}$$

When FFW-NNs are used to approximate arbitrary non-linear functions, it is often helpful to normalize the input, as well as the output data. The reason is that this step moves the data to the unsaturated range of the chosen activation function. Therefore, a normalization to $[-1, 1]$ or $[0, 1]$ increases the performance for the logistic sigmoid or the tangens hyperbolicus, respectively [71].

2.3.2.2 *Training a Multilayer Perceptron Network*

An artificial neural network may either be trained *online* or *offline*. In the *offline* case, the network is presented with a set of training samples, while the training method computes the network weights. Subsequently, the network can be used to compute the output for a given input. *Online* training, on the other hand, means showing the network the samples one by one, while the network weights are updated gradually after each so called instance. Especially the ability to adapt slowly to new situations is helpful in noisy environments where a trade-off between adapting to each (noisy) sample and keeping the past experiences forever has to be found.

Therefore, in *online learning*, the adaptation of the weights is expressed subsequently for each sample, while training is started from random weights. In non-linear regression, the error can be expressed for each sample $X_t = \{x_t, y_t\}$, while in general both x_t, as well as y_t are vectors. Therefore, $x_{t,i}$ and $y_{t,k}$ are the i^{th} input and k^{th} output neuron, respectively. The error can be written as follows.

$$E(X_t, v, w) = \frac{1}{2}\sum_{k}(r_{t,k} - y_{t,k})^2 \tag{2.24}$$

As the output y_k is linear in the output layer weights, each of them can be updated using gradient descent. Here, the weights are updated with a delta that is proportional to the gradient of the error function with respect to the weights

$$\Delta v_{kj} = -\eta \frac{\partial E}{\partial v_{kj}} = -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial v_{kj}} \tag{2.25}$$

$$= \eta (r_{t,k} - y_{t,k}) z_j \tag{2.26}$$

while η denotes the update factor. Due to the activation function the chain rule is used for the first layer weights. Note that in contrast to the output layer weights, the gradient of E with respect to the first layer weights depends on all outputs k.

$$\Delta w_{ji} = -\eta \sum_k \frac{\partial E}{\partial w_{ji}} \tag{2.27}$$

$$= -\eta \sum_k \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_h} \frac{\partial z_h}{\partial w_{ji}} \tag{2.28}$$

$$= \eta \left\{ \sum_k (r_{t,k} - y_{t,k}) v_{kj} \right\} z_j (1 - z_j) x_{t,i} \tag{2.29}$$

Here, the term $\frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_h}$ acts as the unknown error measure in the hidden units. The error of the output units is *back-propagated* to the hidden layer. Hence, the two update rules in Eq. 2.26 and Eq. 2.29 denote the well known *Backprop* (Backpropagation) algorithm [112]. The rules can be easily extended towards updating weights for multiple samples at once, by summing up Δv_{kj} and Δw_{ji} for sets $X = \{x_t, r_t\}_{t=0}^n$.

If samples are presented subsequently, they are often presented multiple times, while each presentation of the full set is called an *epoch*. In this case a smaller update factor η can be chosen, which leads to slower convergence. This is compensated by multiple *epochs* and leads to a better generalization. As each weight is updated for each sample independently from he other samples, update values for subsequent samples might lead to updates that cancel each other out. This effect can be addressed by randomizing the order of the presented samples after each *epoch*.

When parameters are chosen for a given problem, the input and output layer sizes are fixed for the application, as they denote the dimensionality of the inputs and outputs. Therefore, the choices that can be made, are restricted to the number of hidden units, the update factor η, and the number of training *epochs*. If the number of hidden units is chosen too small, the underlying structure of the learned process is not sufficiently captured. If it is chosen too large, the data is *overfitted*. Similar to polynomial fitting with polynomial orders being too high, the learned function captures the noise in all samples leading to poor generalization. The same is true when the samples are presented to often. As there is no general rule for this problem, we can only find a measure for the generalization and choose these parameters by experience until an optimum in the generalization is reached. For this purpose, the samples are split into a training and a validation set by picking

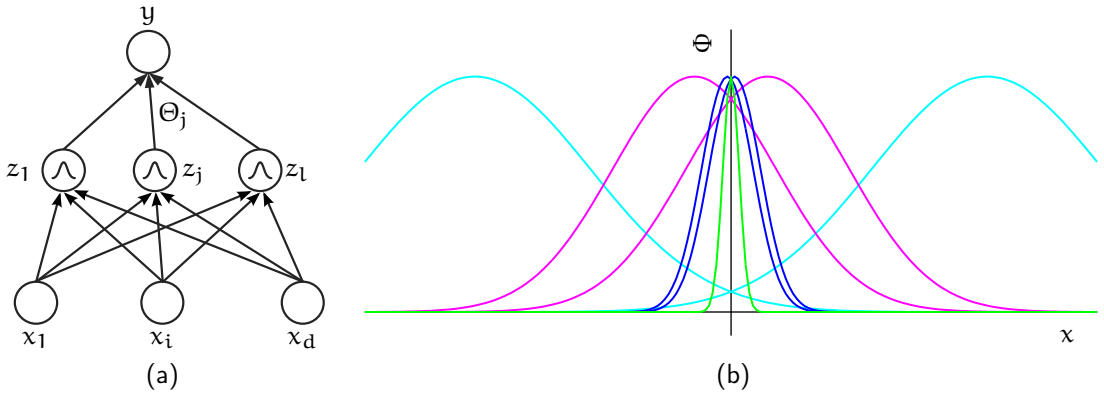


Fig. 2.9: Radial Basis Function Network. The (a) structure shows the input neurons which are connected to a single hidden layer which has radial basis activation functions and a single output neuron, while (b) depicts a family of radial basis functions.

samples for each randomly from the initial set. While the training set is then used for training, an additional set is gained that can be used to measure generalization. For this purpose the network is evaluated for each sample in the validation set. Due to the fact that the network had no knowledge of these samples during training, the MSE over the validation set is a good measure for generalization [8].

2.3.3 *Radial Basis Function Networks*

An MLP is a valuable tool for regression, however, the *Backprop* algorithm exhibits non-linear characteristics. When regression is performed as part of an adaptation process within a control algorithm, it is desirable to use a regression model that is Linear in the Parameters (LP). A versatile regression model that is often used in adaptive control is the RBFN which is a neural network with a slightly different structure from the MLP. This type of network is LP when only a single hidden layer is used (see Fig. 2.9a). It has been proven that an RBFN can approximate any continuous function, given that the number of neurons is sufficient and the parameters of the radial basis functions are appropriate [104].

In contrast to an MLP, in an RBFN there are only weights in the output layer (see Fig. 2.9a), hence simplifying the learning procedure, since the additional step of back propagating the error becomes unnecessary. However, the Radial Basis Functions (RBFs) need to be parameterized manually. An RBF is defined as follows [13],

$$\Phi(x) = \exp - \frac{(x-c)^T \cdot (x-c)}{\sigma^2} \tag{2.30}$$

while c denotes the center of the RBF and σ its standard deviation (see Fig. 2.9b). By learning the weights Θ_j ($j = \{1, \ldots, l\}$), where l denotes the number of neurons, the

neurons are activated for some ranges in the inputs. The full neural network can be expressed mathematically as,

$$y = \Theta^T \Phi(x) \tag{2.31}$$

where $\Theta = [\Theta_1, \cdots, \Theta_l]^T$ denotes the vector of weights.

The advantage of utilizing only weights in the output layer, comes at the price of having to manually choose the parameters for each of the RBFs. Fig. 2.9b shows a family of RBFs with different centers and standard deviations, as would be used in an RBFN. When parameters are chosen, it is important to sufficiently cover the input space, while the individual neurons should at the same time not interfere too much. Similar to MLPs it is also possible to normalize the inputs to some well defined range. The difficulty comes when multi dimensional inputs are considered. The missing input layer weights remove a dimension from the parameter space. Therefore, the information density is less in an RBFN and additional neurons have to be added to cover the input space.

3

THE ANTHROB

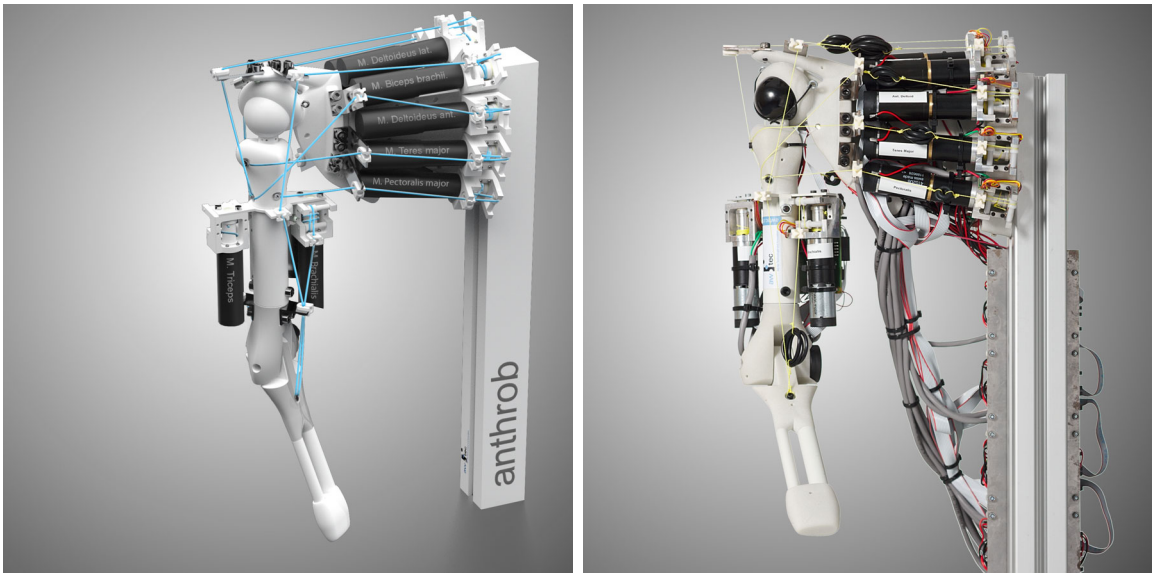
Previously developed musculoskeletal robots showed severe deficits in controllability, due to missing sensor facilities or unavailable dynamic models (see Section 2.1.2). Addressing these drawbacks, a musculoskeletal robot arm was developed. The goal was to create a robot that reflects the challenges in control of musculoskeletal robots, by introducing (1) complex joint types like spherical joints, (2) actuation through multi-articular muscles, (3) compliant actuation and (4) variable lever arms. At the same time it should be possible to extract the dynamics of the skeleton from the CAD models. Hence, an ideal test bed for musculoskeletal control algorithms was developed.

To solve the aforementioned problems with previous anthropomimetic robots, 3d printing techniques were utilized to manufacture the anthropomorphic shapes. The skeletal structure of the arm was developed to have approximately 2/3 of the size of a human male of 1.70 m in height. Therefore, it yields an upper arm with 200 mm and a forearm with 180 mm length (for human reference data see [123]). Furthermore, the robotic arm was designed to carry a load of 500 g.

3.1 SKELETON

The human upper extremity, consisting of shoulder complex, upper arm, elbow joint, forearm and hand has developed to fulfill a wide range of motions. The shoulder complex, comprising the clavicle, the scapula, and the shoulder joint, is often considered to be the most complex joint in the human body. While the clavicle is articulated by a joint towards the rib cage (sternum and first rib) and a joint towards the scapula, the latter contains the glenoid cavity which holds the head of the humerus. This structure serves the purpose of allowing for elevation and depression of the scapula, as well as extending the range of movement of the humerus. The elbow joint, on the other hand, is a revolute joint located between humerus and ulnar and allows for elbow flexion and extension, while the radioulnar articulation is responsible for pronation and supination (rotation around the axis of the forearm) [36].

For the purpose of creating a simplified test bed for control algorithms, it is not necessary to recreate the full range of movements. By fixation of the scapula, as well as the radius, a musculoskeletal robotic arm was developed with four DoF, three in the shoulder joint and one in the elbow (see Fig. 3.2). It allows for the full range of motion in the glenohumeral joint, as well as elbow flexion and extension, while immobilizing shoulder elevation and depression, and the pronation and supination of the forearm. Furthermore, the range of motion of the humerus is limited, as the glenoid cavity remains fixed.



(a) (b)

Fig. 3.1: Anthrob. The (a) rendering is shown next to the (b) final produced robot (design: awtec Zürich)

3.2 ACTUATION

Not all muscles in the human arm could be integrated, nor was it necessary to do so, as the human shoulder complex with a moving scapula and clavicle, as well as the human elbow have more DoF than the skeletal structure considered here. Therefore, only those shoulder muscles were replicated that are involved in glenohumeral movements, reducing the number of muscles of the rotator cuff to eight. Similarly, the omission of the elbow pronation and supination led to a realization of the Biceps (brachii), Brachialis and Triceps (see Fig. 3.3 and Table B.2). To capture the human dynamics of the arm, insertion points were chosen to be as close as possible to their human counterparts. It has to be noted that in the shoulder some muscles are not grown to the bone at specific points, but rather apply force to the bone at larger surfaces (e. g. Infraspinatus [36]). In these cases insertion points were chosen to resemble the functionality of the muscle. Furthermore, the human Triceps has three heads, the long, lateral and medial head, of which the first is bi-articular. The insertion point chosen for the artificial Triceps is equivalent to the medial head, leading to a mono-articular muscle. In the case of the Biceps however, both heads are bi-articular. These could be merged, as well, without loosing functionality.

A modular muscle unit was developed, serving the purpose of a musculoskeletal actuator, including all necessary sensors for control. It comprises a brushed DC motor and gearbox, deflection pulleys for tendon routing, and a spindle that winds up two types of high-performance cords, made from braided ultra high molecular weight Polyethylene (Dyneema® kite line; diameter: 0.6 mm and 1.0 mm), hence contracting the muscle. To

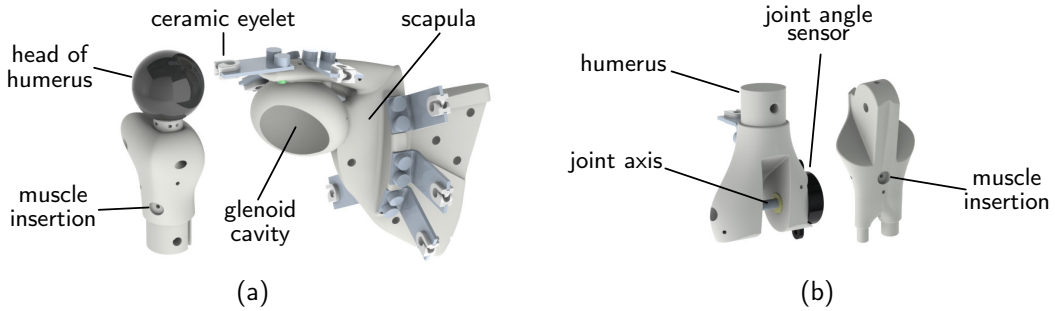


Fig. 3.2: Anthrob Joints. Both (a) the shoulder joint with a fixed scapula and a ball and socket joint between the glenoid cavity and the head of the humerus, as well as (b) the simplified elbow joint with immobile radius and ulnar, are shown.

reduce radial forces on the output shaft of the gearbox a sliding bearing was introduced, supporting the far end of the spindle (see Fig. 3.4). Next to reducing the size, such a modular unit facilitates the producibility and maintainability of the robot.

As the arm should be able to lift its own weight plus an additional weight at the end-effector of 500 g, properly dimensioning the motors was a key factor. However, especially in the rotator cuff, where muscles take different roles depending on the pose, estimating maximum forces was not straightforward. It can be assumed that in the shoulder at least two muscles take the role of holding the arm. Clearly, maximum forces can be observed at full abduction of the shoulder and extension of the elbow joint. Hence, by taking the spindle radii into account, a selection of brushed DC motors and gearboxes was chosen by computing maximum forces and rotational velocities, based on the worst case lever arms. Therefore, two different configurations of brushed Maxon 12 V DC motors and gearboxes were chosen for the shoulder and the elbow joint, respectively (see Table B.1).

To mimic the visco-elastic properties of the human muscle, each actuator was equipped with an Nitrile Butein Rubber (NBR) O-ring, as an elastic element. The properties of NBR O-rings of different sizes were identified by experiments which were conducted to measure the dynamic stress for a step in the strain [139]. It could be seen that the material exhibits a stress relaxation behavior similar to the human muscle [54]. Stiffnesses for the different muscles have been chosen to resemble findings in the human body [44]. For the exact choices, refer to Table B.2.

3.3 SENSORY SYSTEM

In the human body proprioception is achieved by a large set of different sensor modalities that are intelligently fused to obtain the sense of relative position between neighboring parts of the body. This is on the one hand achieved through sensors that measure the state of the muscles, and on the other through—even though not very reliable—joint angle sensors. Biological muscles are equipped with two types of receptors, which are (1) the

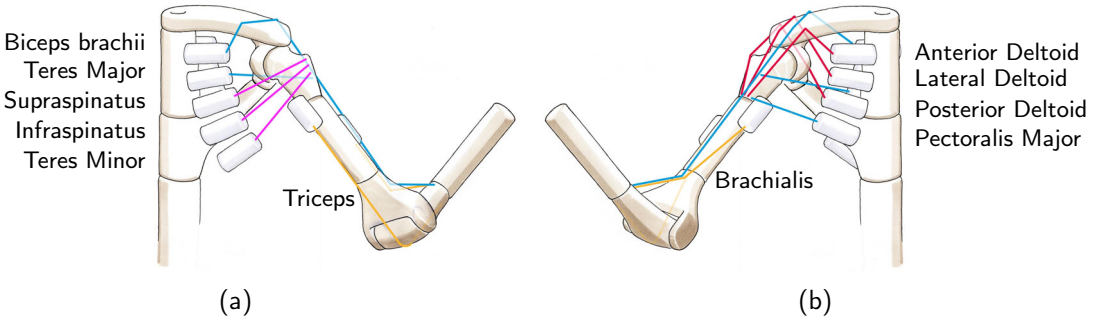


Fig. 3.3: Anthrob Muscles. The 11 muscles are shown in a functional sketch from (a) rear and (b) front, including 8 shoulder muscles, 2 elbow muscles and 1 bi-articular muscle (Biceps brachii). (illustrations: A. Jenter / awtec Zürich)

muscle spindle, which is a sensory receptor encapsulated in the fleshy part of the muscle and (2) the golgi tendon organ, which is located at the attachment point of the tendon to the muscle fibers (see also Fig. 3.6). While muscle spindles are most sensitive to changes in the muscle length, the tendon organs mainly measure the muscle tension [56]. The state of an artificial muscle, based on an electromechanical drive can be fully determined by (1) an actuator position sensor, (2) a current sensor and (3) a muscle force sensor. The first and the second could be easily realized, as motors can be equipped with motor encoders and there are several solutions to measure current which could be integrated into the driver electronics (see Section 3.5). The latter calls for a customized solution, however. One possibility was to only use the motor current sensor and estimate the output torque of the gearbox. While this would lead to good results for higher torques or while the actuator is moving, the static friction in the gearboxes renders this method useless for low torques at motor stall. However, for muscle control, high-quality force measurements are needed. Even though there are strain gauge based solutions available off-the-shelf, they are rather large and hard to integrate into the musculoskeletal setup. Therefore, the most elegant solution was to measure the force directly at one of the muscle insertion points. From an integration point of view it is advantageous to place the sensor on the driving end, as wiring is reduced, even though the measurement is biased by friction in the tendon routing.

The modular motor unit facilitates measuring the tendon force. The tendon is guided by two deflection pulleys and a ceramic eyelet to ensure that the force is measured under the same angle, while being wound up by the spindle (see Fig. 3.4). The two bending beams holding one of the two deflection pulleys are therefore under the stress of the muscle force. This stress can be measured by strain gauges, applied to the bending beams. The unit was carried out as a milled aluminum part, hence leading to a linear stress-strain relationship which can be detected by four strain gauges, measuring twice the elongation due to the stress, and twice the compression on the underside. This way the four resistors

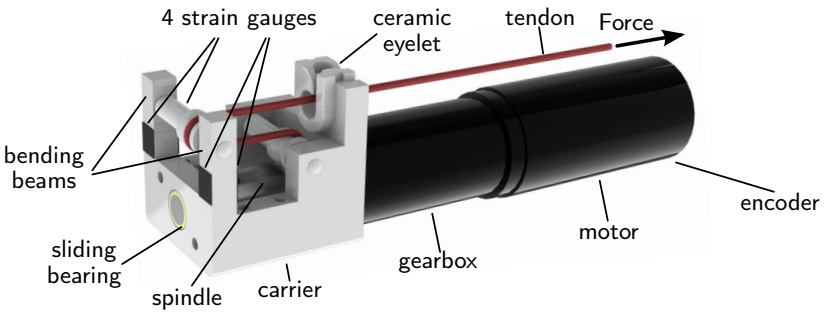


Fig. 3.4: Anthrob Actuator Module. The unit comprises a brushed DC motor, gearbox, and encoder, as well as a carrier for the spindle, which includes the bending beams for measuring the tendon force by means of four strain gauges.

could be interconnected to form a four-arm Wheatstone bridge. A perfectly balanced Wheatstone bridge has an output voltage of zero. It serves the purpose of adding and subtracting the positive and negative changes in the resistance, due to the elongation and compression of the resistors, respectively. Any changes that affect all resistors equally, e. g. temperature, are canceled out automatically [138]. Therefore the output voltage can be easily amplified by an instrumental amplifier. For this purpose custom electronics were developed and produced that could be mounted directly to the force sensor and are able to amplify the Wheatstone bridge output voltage by an amplification gain of 100 or 200, depending on the maximum force produced by the muscle. Furthermore, perfect balancing of the Wheatstone bridge could not be ensured, due to production inaccuracies. Hence, facilities for manual balancing were integrated into the electronics by means of a potentiometer.

Even though human motor control does not directly rely on high quality joint angle sensors, but rather a sense of the lengths of muscles and ligaments are used to get an estimate of proprioception [56], the control algorithms developed in this work rely on reliable joint angle measurements. It would be possible by means of a good model to infer joint angles from actuator positions and tendon forces. However, such a model needs to be learned from samples of the system (see Section 2.3). Unfortunately it is difficult to obtain an analytical model of the muscle kinematics, due to collisions of the tendons with the rigid structures. But even when machine learning techniques are utilized to obtain the muscle kinematics, such a supervised learning approach needs joint angle data to be available during the learning phase. Therefore, the decision was made to rely on direct measurements of the joint angles. Measuring the joint angle of the elbow joint was realized in a straightforward manner, as it comprises a single DoF, by including a potentiometer measuring the rotation of the elbow axis. For a spherical joint this is rather difficult as shown by Nakanishi et. al. [89]. The easiest way was to use an extrinsic motion capture unit to obtain the missing information. Therefore, a stereo-vision based approach, developed by Wittmeier [139, 142], was utilized that is able to capture the full

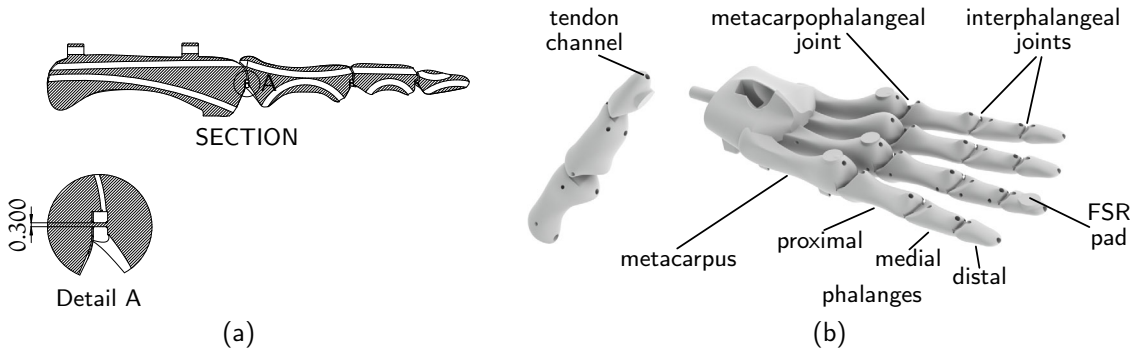


Fig. 3.5: Anthrob Hand. The (a) cross-sectional view of the forefinger is shown, depicting the solid state hinge joint, as well as (b) the two parts of the printed hand, the thumb and the metacarpus with the four fingers.

pose of bodies equipped with infrared markers. By adding markers to the scapula, as well as the humerus, the rotation of the shoulder joint is extracted [29].

3.4 HAND

The previously mentioned hands have been developed to offer the best possible dextrous manipulation capabilities [74, 117], leading to highly complicated designs with many DoF and up to 20 actuators. However, the scope of this work was to design an anthropomimetic robot arm as a platform for tendon-driven control development. Therefore, only some key features were realized for the hand. These were (1) to be able to actively open and close the hand and (2) apply a power grip utilizing flexible elements to automatically adapt to different objects. Finally, the hand should be (3) as light as possible and (4) easily producible without the need for complex assembly steps.

The bones were inspired from the human hand and replicate each of the phalanges, including the metacarpus (see Fig. 3.5b). By fully exploiting the technology of 3d printing it was possible to produce the hand only from two parts and include printed channels for tendons and wires. This was realized by developing solid state hinge joints (see Fig. 3.5a). In this type of joint, the two movable parts are only connected by a thin film (0.3 mm) of material. The thickness together with the choice of Polyamid 2200 (PA-2200) as a constructive material makes this thin film bendable and also durable enough to withstand the forces when objects are grabbed. Special care had to be taken to apply the printed layers in the pane of the film. If they were applied perpendicular to the film, the material would break. Due to the simplified grasping requirements, it was possible to reduce the number of actuators to two, one for closing and one for opening the hand. For this purpose two DC motors were mounted on the forearm, actuating two tendons. In the base of the hand these split into five separate elastic elements, so that each finger is able to adapt to different objects. From there two tendons run down each finger, actuating each of the

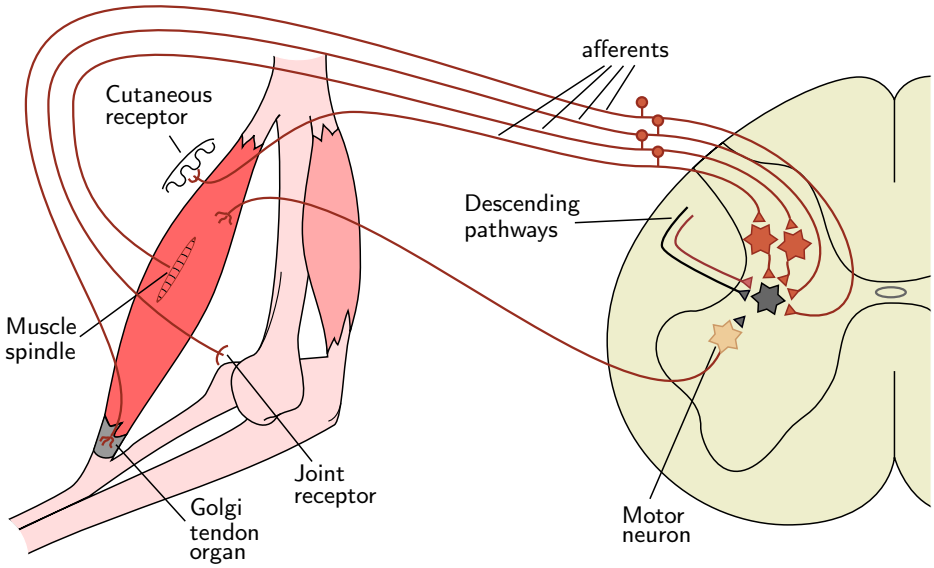


Fig. 3.6: A motor nucleus is a cluster of motor neurons in the human spinal cord (or brain stem). It is responsible for muscle control and is directly connected to the related receptors (adapted from [56]).

metacarpophalangeal and the two interphalangeal joints at the same time (see Fig. 3.5b). To execute a power grip it had to be ensured that the joints close in the proper order from proximal to distal. Otherwise the object would not be enclosed by the fingers. This can be either realized by different thicknesses of the joint film, or by different lever arms of the tendons towards the center of rotation of the joint. It was chosen to do the latter (see Fig. 3.5a).

Even when only a power grip is applied, sensing the grip force is crucial. The force sensor utilized for the arm muscles could be easily adapted to the smaller finger muscle actuators. However, the grip force itself can be measured directly at the finger tips, by integrating Force Sensitive Resistors (FSRs) in strategic places on the hand (see Fig. 3.5b). To maximize the FSR sensitivity and improve the friction between the finger tip and the grabbed object, the thin material of the sensor itself was embedded between two layers of rubber material.

3.5 CONTROL ARCHITECTURE

A control architecture was developed and implemented in all the robots of the ECCE family, as well as the Anthrob. It realizes a biologically inspired hierarchy, by splitting the tasks into voluntary high-level behaviors and involuntary low-level control. Special attention was given to scalability, as robots with up to 50 artificial muscles should have been possible to control.

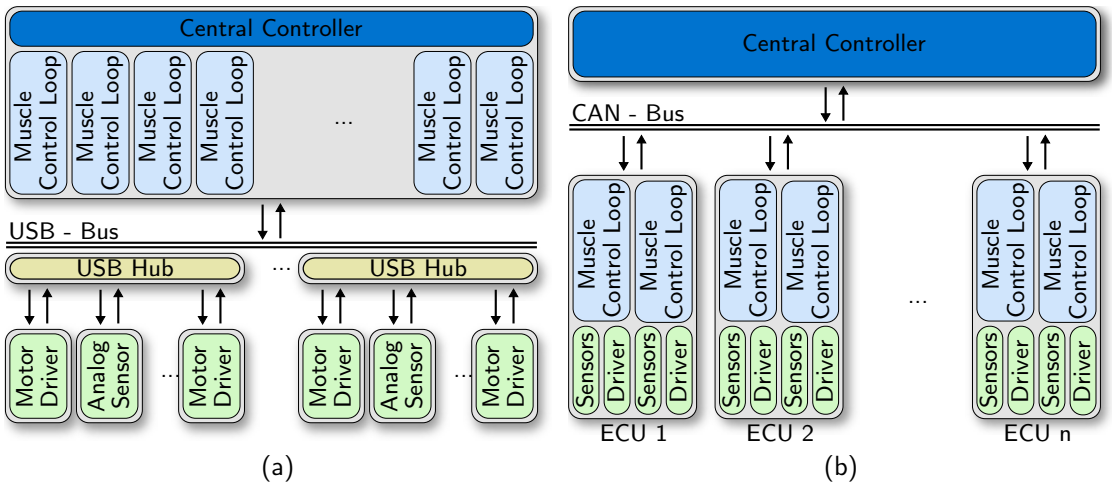


Fig. 3.7: Control Architecture. In a centralized control architecture (a) like it is used for Kotaro [84] all control is run on a single controller. The control architecture proposed here (b) distributes fast local control loops into the robot's limbs, while communication with the central controller is accomplished via a CAN bus system.

Typically, robot control is done using a centralized control scheme, where all sensors and actuators are connected to a single controller. The control algorithm that is being executed on this central controller fetches the sensor values and calculates the actuation for all joints in a single step, which delivers the highest flexibility when the number of DoF is limited. With the number of muscles this approach becomes increasingly complex. In the human body on the other hand, motor control is organized in a hierarchy. While most of the low-level control takes place in the spinal cord and brain stem, voluntary motor control commands are issued by the fore brain. Typically a muscle is controlled directly by a set of motor neurons in the spinal cord that form a motor nucleus [56]. The fore brain can issue commands to the motor nuclei through descending pathways (see Fig. 3.6). The existence of the motor nuclei shows that in the human body the control is highly distributed, where fast low level control is conducted as close to the muscles as possible and the higher levels of (voluntary) movement control communicate with the muscles through the distributed units (motor nuclei). Voluntary reaction times range from 60 ms to 120 ms and can get as low as 40 ms for reflexes [56], which shows that latencies in the human body are actually much higher than in today's robots where control algorithms run at frequencies of 1 kHz to 2 kHz or higher. Still a human is capable of achieving high-speed motions through FFW control, by exploiting the intrinsic dynamics of the body and nervous system.

A robust control architecture was designed, which reduces the complexity of the control task. One way to do this was to stay close to the human archetype and distribute processing units (motor nuclei) around the robot's body to be as close to the sensors and actuators as possible. Each of the boards is connected to a central controller (fore brain)

via a communication bus and therefore only a single bus link plus electric power needs to be routed to the boards. This reduces the cabling, since power and information can be distributed in a tree-like manner. In [84], Mizuuchi et al. proposed a control architecture which was implemented for the robots at the Jouhou System Kougaku Laboratory of the University of Tokyo (JSK). Here, sensor data and motor commands are transferred via the Universal Serial Bus (USB) to distributed nodes. In this setup, however, there is no processing in the distributed nodes. The control algorithm itself is still centralized.

In contrast, the architecture developed here was designed to run fast local control loops, like force, current or position control on the distributed nodes. Additionally sensor preprocessing or fusion, like filtering can be implemented. The required bandwidth for communication could be estimated by assuming the transfer of three sensor values (at 16 bit) and one motor control command (at 16 bit) per actuator at a high-level control frequency of 200 Hz. Additionally an extra 4 bit for the control type was reserved. A limit of 50 muscles was assumed to estimate the following upper bound for the bandwidth BDW.

$$\begin{aligned} \mathrm{BDW} &= 50 \cdot ((4\,\mathrm{bit} + 16\,\mathrm{bit}) + 3 \cdot 16\,\mathrm{bit}) \cdot 200\,\mathrm{Hz} \\ &= 680\,\mathrm{kbit/s} \end{aligned} \tag{3.1}$$

This relatively low bandwidth allows for utilization of the simpler CAN bus instead of faster buses like USB, for instance. This comes with the advantage that CAN is only a two wire serial bus and unlike USB no hubs are required, facilitating the task of wiring. For the purpose of distributing the control nodes, custom electronics were developed. Instead of controlling just one, size constraints led to the decision of allowing for the control of two actuators per Electronic Control Unit (ECU). Each of the developed ECUs features a STMicroelectronics STM32F microcontroller, incorporating a 72 MHz ARM Cortex-M3 processor, several Analog-to-Digital (AD) converters and an integrated CAN interface, as well as power electronics for two motors. The motors are controlled by Pulse-Width Modulation (PWM), using two full H-Bridges. Direct current feedback is given by an integrated hall-effect-based current measurement unit in the motor loop.

To achieve a high level of robustness, the firmware was developed to implement a Finite State Machine (FSM) which can be easily controlled from the central processing unit (see Fig. B.1), using a custom communication protocol. This protocol was based on raw CAN and defines messages needed to initiate transitions of the FSM. Several messages require a reply from the distributed node to be able to determine communication or controller failure. At the same time, a heartbeat is broadcasted to all nodes, allowing for safe shutdown, in case of communication failure. Whenever a node detects an error it transits into a failure state and stops replying to messages (fail-silent behavior [108]). As each processing unit handles the control of two muscles, the FSM is instantiated twice on each ECU. In the *On* state the low-level control law is executed at the fixed frequency of 1 kHz.

4

MODELING MUSCULOSKELETAL ROBOTS

In order to develop controllers for any system, the first step is to find a model that explains the behavior of a system over time for given inputs. By modeling we mean to capture the necessary aspects of the system dynamics in a set of differential equations. Kinematics describes the part of dynamics which studies the motion of particles and rigid bodies without taking the forces which cause motion into account. Purely kinematic models are therefore models which describe the geometry of the motions. Apart from the kinematics, the field of dynamics includes the kinetics which cover how the motion of bodies is caused by applied forces [79]. In robotics, a model is either purely kinematic, only covering the geometry of the motions, or it is a full dynamic model which includes both the kinematics and the kinetics.

When models are obtained, it is possible to define them in several different ways, depending on the choice of system states. In the last years, research in robotic control has focussed on flexible joint robots. It will be shown in the following that the models used for these controllers share several properties with musculoskeletal robots that make it possible to also adopt some of the control concepts. For flexible joint robots, two separate system models can be found in the literature. The first one is based on the joint torque and is written as follows [5].

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_G(q) = \tau + DK^{-1}\dot{\tau} + \tau_{ext} \tag{4.1}$$

$$J_T \cdot \ddot{\theta} + \tau + DK^{-1}\dot{\tau} = \tau_{act} - \tau_f \tag{4.2}$$

$$\tau = K(\theta - q) \tag{4.3}$$

Here, Eq. 4.1 expresses the relation of the joint coordinates and the joint torques at the presence of external torques τ_{ext} that act on the robot, e. g. by interaction with the environment. Eq. 4.2 expresses the actuator dynamics with the possibility to introduce joint and actuator friction as an additional term τ_f, while τ_{act} denotes the actuator output torque. Finally, the third expression Eq. 4.3 relates the two previous equations. K and D are positive definite diagonal matrices, denoting the joint stiffness and damping coefficients, respectively. Hence the model is subdivided into a model for the stiff robot links, the actuator dynamics, and a link between the two.

The second system model is solely based on the positional coordinates of the joints q and the actuator positions θ. The following more compact representation can be obtained

by substituting the joint torque τ, and its first order derivative, by Eq. 4.3. Often, the joint damping D is neglected in these cases [95].

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_G(q) = K(\theta - q) + D(\dot{\theta} - \dot{q}) + \tau_{ext} \tag{4.4}$$

$$J_T \cdot \ddot{\theta} + K(\theta - q) + D(\dot{\theta} - \dot{q}) = \tau_{act} - \tau_f \tag{4.5}$$

Even though this latter model is mathematically equivalent and more compact than the first, it can be seen that Eq. 4.1 - 4.3 can be subdivided in three subsystems. In the following, these are used as a starting point for dividing the dynamics of musculoskeletal robots. In line with this subdivision, three submodels are found. Starting from (1) a model of the skeletal dynamics (see Section 4.1), to (2) a model of the muscle dynamics (see Section 4.2), and finally (3) a link between the two, i. e. the muscle kinematics (see Section 4.3), is obtained. Subsequently the full system dynamics, assembled from the three submodels, will be presented in Section 4.4.

4.1 SKELETAL DYNAMICS

4.1.1 *Kinematics*

The kinematic description developed in the following is applicable to any robot that can be expressed as a kinematic chain, comprising stiff links and interconnecting joints (see Fig. 4.1). The kinematics of such a robot yield the *pose* (i. e. the positional coordinates), but also the velocity and all higher order derivatives [119], of each of the robot links with respect to the joint coordinates and its derivatives. For the links this implies that there is no bending, leaving the link transformation matrix T_L between two joints to be constant. The movement of the joints is fully constrained to a defined set of DoF, making the joint transformation matrices T_J a function of the respective joint coordinates. For each joint the joint coordinates contain the position of the unconstrained modes. In case of a revolute joint this is simply the angular position and for a prismatic joint, the linear displacement. We write the full vector of joint coordinates q as a vector containing all joint positions in defined order.

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \tag{4.6}$$

In this work the transform of the i^{th} joint frame f_{Ji} and the moving joint frame $f_{J'i}$ is defined as ${}^0T_i$ and ${}^0T_{i'}$, respectively. We usually distinguish between *forward* and *inverse* kinematics of a robot. The first expresses simply the end-effector position and orientation in the base frame, subject to the joint coordinates q (see Fig. 4.1).

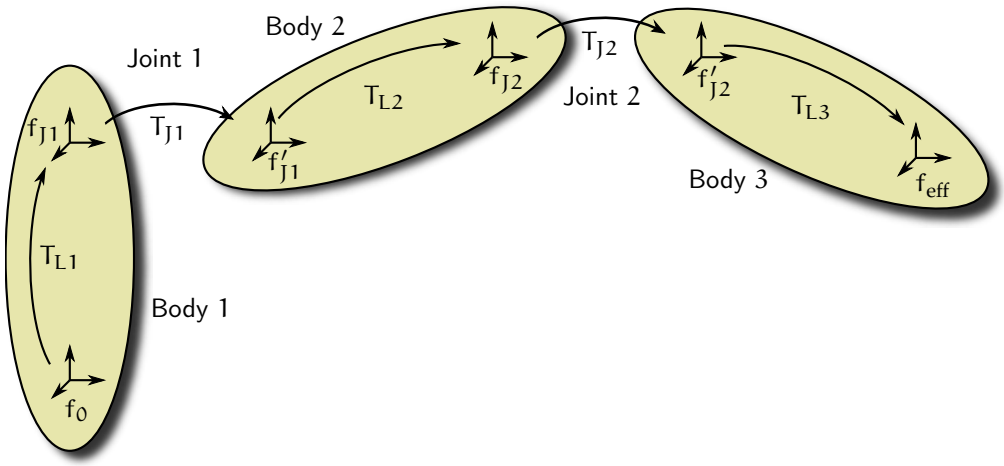


Fig. 4.1: Robot Kinematics. The definitions include transforms for each link and joint between the base frame f_0 and the end-effector frame f_{eff}.

$$^0T_{eff}(q) = \prod_{J,L} T(q) \tag{4.7}$$

Another possibility to map the operational space to the joint space, is to use instantaneous kinematics. It states a linear relationship between the rates of changes of the operational pose and the joint positions at a certain joint configuration, By differentiating $^0T_{eff}(q)$ with respect to these joint angles we obtain a Jacobian matrix which is commonly referred to as the "joint Jacobian" or simply "Jacobian" and yields the *forward* instantaneous kinematics

$$\dot{x} = J(q) \cdot \dot{q} \tag{4.8}$$

for the time derivatives of the joint coordinates q and the operational space coordinates x. In cases where $J(q)$ is square, or in other words the number of operational DoF is equal to the joint DoF, it is possible to directly invert $J(q)$ to gain the *inverse* instantaneous kinematics, given the Jacobian is not singular.

The joint Jacobian offers a dual relationship. It relates motion in the joint space with motion in the operational space (see Eq. 4.8), but due to its geometric properties it also gives a mapping from the operational space forces F to the joint torques τ. The latter relationship can be obtained, utilizing the *principle of virtual work* [35]. We state equality of the instantaneous amount of work, applied in both spaces.

$$F^T \cdot \partial x = \tau^T \cdot \partial q \tag{4.9}$$

Expanding by $\frac{1}{\partial t}$ and inserting Eq. 4.8 we get

$$F^T \cdot J(q) \cdot \dot{q} = \tau^T \cdot \dot{q} \tag{4.10}$$

which solving for τ, finally yields the relation between joint torques and end-effector forces.

$$\tau = J^T(q) \cdot F \tag{4.11}$$

The joint Jacobian is a purely geometric property, relating both joint space to operational space velocities, as well as operational space forces to joint torques.

4.1.2 *Dynamics*

The kinematic descriptions that can relate the joint movements, expressed in joint coordinates, with the movements of the robot links, neither capture the effect of forces that act externally on the robot (contact forces), nor can they express the joint torques necessary to generate certain movements. Rigid body dynamics describe a relationship between the forces and the motion coordinates. For robotics, the mapping of actuation forces and resulting joint acceleration for a given state of the robot, is of interest, as it allows both for simulation, as well as control of robots. Generally, a distinction between inverse and forward dynamics is made. While the first computes the necessary joint actuation forces for a given trajectory, the latter determines the joint acceleration for given actuation forces. Forward dynamics are mostly used in simulation of robots, whereas the inverse dynamics are utilized in robotic control laws to e. g. linearize the dynamics (see Section 2.2.1 and [119]).

For conventional robots the inverse dynamics relate the joint torques τ with the joint accelerations $\ddot{q}$, for a state of the robot, given by $q, \dot{q}$. It can be expressed in the so called canonical form,

$$\tau = H(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_G(q) \tag{4.12}$$

while $H(q)$, the mass matrix, is an $n \times n$ symmetric, positive definite matrix, expressing the robot inertia, $C(q, \dot{q})$ is a $n \times n$ matrix, which accounts for Coriolis and centrifugal effects, and $\tau_G(q)$ is the vector of gravity terms. The equation of motion can be extended to include additional dynamic effects like friction or external forces.

It can be shown that $\dot{H}(q) - 2C(q, \dot{q})$ is skew-symmetric such that

$$z^T(\dot{H}(q) - 2C(q, \dot{q}))z = 0 \tag{4.13}$$

for any $n \times 1$ vector z. For control it is important to note that both $\|C(q, \dot{q})\|$ as well as $\|\tau_G(q)\|$ have an upper bound, which in the first case is a function of $\dot{q}$ [119].

Even though muscles exhibit significant compliance, the skeleton of musculoskeletal robots can, in most cases, be assumed to be rigid links between the joints. This assumption will hold for the operational range of the robots, as the links have been designed to show little strain under the loads considered. In any case, the compliance of the muscles is higher by a manifold and therefore deformation of the links will be neglected throughout this work.

Hence a model of the skeletal dynamics can be built upon the canonical form of the inverse dynamics in Eq. 4.12. The equation of motion holds for all joint types, however it is generally used for single DoF joints. Like the human body, musculoskeletal robots make use of revolute, as well as spherical joints. While the positional joint coordinate of a revolute joint is simply the angle, the three dimensional rotation of a spherical joint can be expressed in several different ways. Due to its lack of singularities, the unit quaternion description of rotation was chosen to describe the rotation of all spherical joints (see also Section A.2). This comes with two challenges, which are (1) the fact that unit quaternions have a higher dimensionality than the number of DoF and (2) that joint velocities and accelerations cannot be expressed as the derivatives of the positional coordinates directly. Both are addressed by replacing the positional joint coordinate vector q by α, containing the quaternion representation of all spherical joints, as well as the angular representation of all revolute joints, leading to a modified joint space dynamic equation.

$$\tau = H(\alpha)\ddot{q} + C(\alpha, \dot{q}) + \tau_G(\alpha) \tag{4.14}$$

Therefore, α exhibits a higher dimensionality than q and a relationship between the derivative of α and the rotational velocities $\dot{q}$ needs to be obtained. At this point it has to be defined, whether joint velocities and accelerations of spherical joints are to be expressed in the fixed coordinate frame of the joint f_{Ji}, i. e. the coordinate frame of the link closer to the base, or in the moving coordinate frame f'_{Ji} (see Fig. 4.1). In robots with only revolute and prismatic joints, as is the case in most common robots, this is not necessary, as the velocities are equal in both frames. While both definitions are valid, they have to be consistent throughout the model. Hence, for Eq. 4.14 to be valid, the joint torques τ have to be defined in the same coordinate frames as the velocities and respective accelerations. In this work these quantities are expressed in the moving coordinate frame. Therefore, henceforth $\dot{q}$, $\ddot{q}$ and τ denote the joint velocities, acceleration and torques, where the respective i^{th} entry is expressed in f'_{Ji}.

The derivative of a quaternion $\dot{Q}$ as a function of the corresponding rotational velocities $\vec{\omega}'$ in the moving coordinate frame f'_{Ji} is stated by the matrix $U(Q_i)$ (see Section A.2). Therefore, a matrix $A(\alpha)$ can be defined, as a diagonal matrix,

$$A(\alpha) = diag(\tilde{A}_i) \qquad \tilde{A}_i = \begin{cases} 1 & \text{if joint } i = \text{revolute} \\ \frac{1}{2}U(Q)^T & \text{if joint } i = \text{spherical} \end{cases} \qquad i = \{1, \dots, N\} \tag{4.15}$$

while $\tilde{A}_i$ is defined by the joint type and N is the number of joints. Therefore, A denotes the relation between the derivative of the positional coordinate vector $\dot{\alpha}$ and the joint velocities $\dot{q}$

$$\dot{\alpha} = A(\alpha) \cdot \dot{q} \tag{4.16}$$

Obviously, the definition of $A(\alpha)$ can be easily extended to cover any other joint type, however, throughout this work, only revolute and spherical joints are considered. Due to

the fact that the mapping $U(Q_i)$ is orthogonal (see Section A.2), the inverse of $A(\alpha)$ can be defined as follows.

$$\dot{q} = A(\alpha)^{-1} \cdot \dot{\alpha} = \mathrm{diag}(\tilde{A}_i^T) \cdot \dot{\alpha} = A^T(\alpha) \cdot \dot{\alpha} \tag{4.17}$$

An equation of motion of the skeleton in the canonical form (Eq. 4.14) can be computed, using the Newton-Euler Algorithm. This well known rigid body dynamics algorithm allows for efficient computation of the inverse dynamics. Extensions towards using quaternions as positional coordinates can be included into any implementation of the Newton-Euler Algorithm by modifying xjcalc and pcalc to use quaternions as input and produce the spatial transforms X_J and Φ_i, respectively. A detailed description of the algorithm is given in Section A.4.

Implementations of the Newton-Euler Algorithm fall into two categories which produce either (1) closed-form solutions by symbolic computation, or (2) numeric online evaluation at an operation point, specified by the system state α, $\dot{q}$. Since the complexity of the algorithm is $\mathcal{O}(n)$ (see Section A.4), the computational cost of online evaluation is manageable. As an additional advantage, the description of the input model for numeric online evaluation of the Newton-Euler algorithm can be stated in a highly generic manner. The *Robotics Library*[1] implements this solution and is used throughout this work for computations that involve the skeletal dynamics. The dynamics algorithm takes an Extensible Markup Language (XML) description of the robot that covers the kinematic chain from base to end-effector by stating joint types and transformations in between the joints. At the same time dynamic properties are assigned to the links of the robot by stating the center of mass, the inertia tensor, as well as the mass for each body (see Listing B.1). This data can be extracted from the CAD models of the robot.

4.2 MUSCLE DYNAMICS

For control purposes it is inevitable to have a dynamic model of the actuator. The muscle dynamics obtained in this section are based on the standard model for brushed DC motors [135]. Next to brushed DC motors, the usage of brushless DC motors has grown more common in the last years. Even though the methods provided throughout this work, were developed for the example of brushed DC motors, an extension to brushless motors is straightforward. The brushed DC motor model can be subdivided into the electrical part and the mechanical part of the motor and the mechanical model of the gearbox. First, the electrical part, can be obtained by applying Kirchhoff's loop rule to the armature loop (see Fig. 4.2),

$$v_A = R_A i_A + L_A \frac{di_A}{dt} + v_{emf} \tag{4.18}$$

1 The Robotics Library (RL) by Markus Rickert covers platform independent implementations of available robotics algorithms for forward and inverse kinematics and dynamics, as well as path planning and collision detection [109]. In this work version 0.6 was utilized with extensions for spherical joints to compute the inverse dynamics.

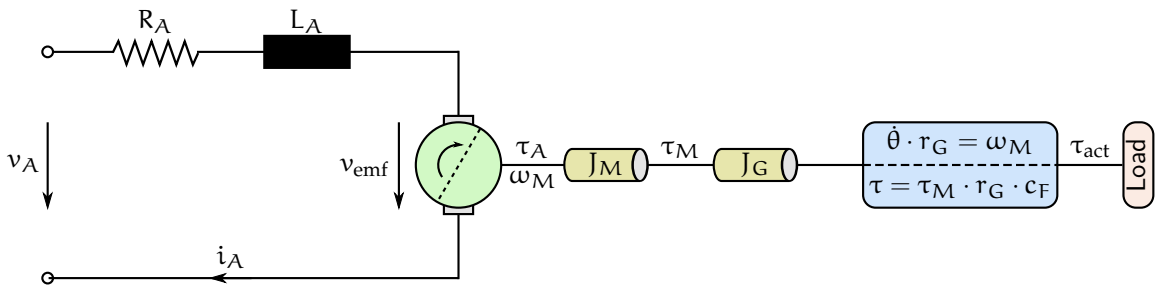


Fig. 4.2: Actuator model. It comprises a model of the electrical and the mechanical part of the brushed DC motor, as well as a model for the gearbox.

while v_A is the armature voltage, R_A and L_A are the resistance and inductance, respectively, i_A is the current and v_{emf} is the back Electro-Magnetic Force (EMF). In robotics, most commonly, motors with permanent field magnets are used. For this type of motor, the flux Φ is assumed to be constant and the EMF constant c_{emf} equal to the torque constant c_τ. Therefore, the electrodynamics can be modeled by a single linear motor constant c_M

$$v_{emf} = c_{emf}\Phi \cdot \omega_M = c_M \cdot \omega_M \tag{4.19}$$

$$\tau_A = c_\tau \Phi \cdot i_A = c_M \cdot i_A \tag{4.20}$$

with τ_A as the anchor torque and ω_M as the rotational velocity of the motor. This linear model is true in the case of good quality motors that are used within specification, however, at very high currents Eq. 4.20 does not hold due to saturation effects in the metal parts. If motors are in these operation ranges, a different model has to be found. In this work, it is assumed that motors are used within specification, where Eq. 4.20 holds. Finally, the generated torque τ_A is driving the motor inertia J_M and the output torque of the motor τ_M.

$$\tau_A = J_M \dot{\omega}_M + \tau_M \tag{4.21}$$

While there is a standardized model for brushed DC motors [135], gearboxes are modeled in different ways depending on the accuracy that needs to be achieved. The most basic model includes only the gearbox ratio r_G which relates the output torque and velocity to their respective inputs. Ignoring friction is an oversimplification, as even good quality gearboxes have maximum efficiencies between 40 % and 85 %, depending on the number of stages. Therefore, a model of the friction is necessary. However, detailed modeling of the gearbox friction losses can lead to highly complicated models [100]. There are even models that consider backlash and hence add hysteresis, however in the following, the models are kept as simple as possible. The degree of efficiency is another form of describing a torque dependent friction component, which can be explained by the Coulomb friction (see Section A.5) between the teeth of the gears, increasing with the transmit-

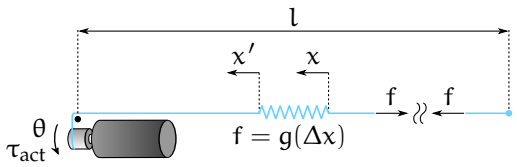


Fig. 4.3: Definition of muscle coordinates. The actuator position θ is linked to the muscle length l through the expansion of the elastic element Δx which is determined by the muscle force f and the stress-strain relationship of the elastic material.

ted load. The frictional loss is further increased by a viscous friction component, hence providing the following model for the gearbox velocity and torque relationship

$$\dot{\theta} = \frac{1}{r_G}\omega_M \tag{4.22}$$

$$\tau_{act} = r_G \cdot (\tau_M - J_G\dot{\omega}_M) - \mu_c f_n - \mu_v \cdot \omega_M \tag{4.23}$$

with $\dot{\theta}$ as the actuator velocity and τ_{act} as the actuator torque, which is equal to the load torque (see Fig. 4.2). Hence θ is the actuator position (see Fig. 4.3). If the assumption is made that the normal force on the teeth of the gears f_n is linear in the transmitted load, a torque dependent friction factor c_F can be defined for the gearbox, simplifying Eq. 4.23.

$$\tau_{act} = r_G \cdot c_F \cdot (\tau_M - J_G\dot{\omega}_M) - \mu_v \cdot \omega_M \tag{4.24}$$

The models of the two actuator components can be combined into one state space equation describing the dynamics of the actuator,

$$\frac{d}{dt}\begin{bmatrix} i_A \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\frac{R_A}{L_A} & -\frac{c_M r_G}{L_A} \\ \frac{c_M}{r_G J_T} & -\frac{\mu_v}{c_F r_G J_T} \end{bmatrix} \cdot \begin{bmatrix} i_A \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} \frac{1}{L_A} & 0 \\ 0 & -\frac{1}{c_F r_G^2 J_T} \end{bmatrix} \cdot \begin{bmatrix} v_A \\ \tau_{act} \end{bmatrix} \tag{4.25}$$

while $J_T = J_M + J_G$. This model is linear, as long as the efficiency is modeled linearly. However, the torque dependent friction component changes sign when the drive train switches from the motor driving the load to the load driving the motor. This happens frequently when accelerating and decelerating and can be modeled as follows,

$$c_F = \begin{cases} c_{FF} & \text{if} \quad i_A \cdot \omega \geqslant 0 \\ \frac{1}{c_{FF}} & \text{if} \quad i_A \cdot \omega < 0 \end{cases} \tag{4.26}$$

while c_{FF} is the torque dependent friction factor of the gearbox in forward mode. This frictional behavior of the model is obviously non-linear, but it can be easily used in simulation, and even be extended to model stick-slip behavior [140].

The muscles of the musculoskeletal robots, considered in this work, consist of an electromechanical DC motor and gearbox, a tendon made from kite line and an elastic element (see Fig. 4.3). The muscle dynamics can be obtained by extending the actuator

model by a spindle that winds up the tendon and the elastic element. The spindle is responsible for transferring the rotational movement and torque of the motor into a linear movement and force. In cases where the tendon is wound up on itself in an uncontrolled manner, models for the spindle can become highly complex. This effect can only be modeled if constructive measures are taken to control the tendon winding. For the robots used in this work, the maximum tendon length that is wound up is small enough for this effect to be negligible. A simple model can be expressed as follows,

$$x' = R_S \cdot \theta \tag{4.27}$$

$$\tau_{act} = R_S \cdot f \tag{4.28}$$

where R_S denotes the spindle radius, f the muscle force and x' the linear displacement of the tendon (see Fig. 4.3). Neglecting any damping that might be present, the most general model of the elastic element is described by the stress-strain relationship of the material used.

$$f = g(\Delta x) \qquad \text{with} \quad \Delta x = x' - x \tag{4.29}$$

Introducing Eq. 4.27 and replacing the linear coordinate x by the muscle length l and an initial length l_0 which is the length of the muscle at $\theta = 0$ and zero expansion, the following expression is obtained.

$$f = g(R_S\theta + l - l_0) \tag{4.30}$$

In order to model the full system, it is necessary to express the change in force with respect to the other system states. For this purpose the time derivative of the force is found.

$$\dot{f} = \frac{\partial g}{\partial \Delta x} \cdot (R_S\dot{\theta} + \dot{l}) \tag{4.31}$$

In cases, where the flexible element can be assumed to have a linear stress-strain relationship, this expression can be simplified,

$$\dot{f} = K \cdot (R_S\dot{\theta} + \dot{l}) \tag{4.32}$$

where K is the diagonal matrix of spring constants. By taking the motor current, the motor velocity and the muscle force as system states, a full state space model for the linear muscle dynamics can be found for a single muscle.

$$\frac{d}{dt}\begin{bmatrix} i_A \\ \dot{\theta} \\ f \end{bmatrix} = \begin{bmatrix} -\frac{R_A}{L_A} & -\frac{c_M r_G}{L_A} & 0 \\ \frac{c_M}{r_G J_T} & -\frac{\mu_v}{c_F r_G J_T} & -\frac{R_S}{c_F r_G^2 J_T} \\ 0 & K \cdot R_S & 0 \end{bmatrix} \cdot \begin{bmatrix} i_A \\ \dot{\theta} \\ f \end{bmatrix} + \begin{bmatrix} \frac{1}{L_A} & 0 \\ 0 & 0 \\ 0 & K \end{bmatrix} \cdot \begin{bmatrix} v_A \\ \dot{l} \end{bmatrix} \tag{4.33}$$

With some restriction, this model can even be used for finding a controller, when the elastic element is non-linear, by linearizing non-linear stress-strain relationship around an operating point.

$$K(\Delta x) = \frac{\partial g}{\partial \Delta x}(\Delta x) \tag{4.34}$$

A measure that is often taken to simplify the system model, is to neglect the effects of the motor inductance L_A. This removes the current as an additional state which depending on the control frequency is often not directly controllable. In this case, the non-linear dynamics can be calculated to be.

$$\begin{aligned} \dot{f} &= \frac{\partial g}{\partial \Delta x}(R_s\dot{\theta} + \dot{l}) \\ \ddot{\theta} &= \left(-\frac{\mu_v}{r_G c_F J_T} - \frac{c_M^2}{R_A J_T}\right)\dot{\theta} - \frac{1}{r_G^2 c_F J_T} R_s f + \frac{c_M}{R_A J_T r_G} v_A \end{aligned} \tag{4.35}$$

Hence, two different models were found that cover the muscle dynamics. First, a linearized model of the full state space system was developed. The non-linear dynamics were shown for the reduced system states by neglecting the effects of the motor inductance.

The models stated in this section were developed for single actuators. In the following, the effects of multiple muscles on the skeleton is considered. Therefore, the actuator states are redefined to be vectors denoting the states of all actuators, i. e. θ, τ_{act}, f and l are in $\mathbb{R}^m$, where m is the number of muscles, and denote the actuator positions, the actuator torques, the muscle forces, and the muscle lengths respectively. The dynamics of a single muscle remain unchanged by this redefinition and are evaluated separately for each muscle.

4.3 MUSCLE KINEMATICS

When we talk about muscle kinematics, we mean a geometric model of the muscle routing. The goal is to find a description of how the muscles act upon the skeleton, defining the interaction of the muscle and the skeletal dynamics. In flexible joint robots, Eq. 4.3 takes this role by relating the angular position of the actuator and the joint angles with the joint torque. This can be applied for tendon-driven robots with the N configuration, as the tendons apply torques in both directions and the tendon can be modeled with a given stiffness. However, starting from the $N+1$ configuration, a different route has to be taken, since there is no one-to-one mapping of tendon forces to joint torques. Here, the muscle forces have to be expressed in the joint space.

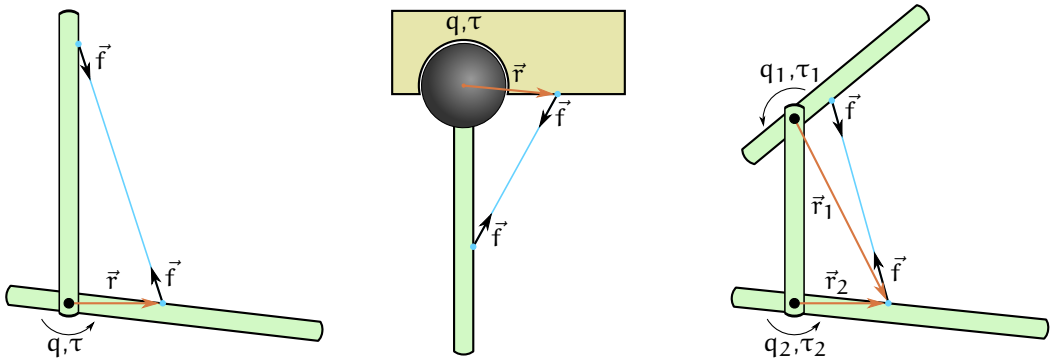


Fig. 4.4: The muscle kinematics are determined by the routing of the tendons. The joint torque can be computed from the three dimensional muscle force vector $\vec{f}$ and the point of actuation of the muscle $\vec{r}$. Three exemplary cases are shown, (a) a uni-articular muscle on a revolute joint, (b) a muscle acting on spherical joint, and (c) a bi-articular muscle spanning two revolute joints.

4.3.1 *Muscle and Joint Space Mappings*

For robots with point to point tendon routing and fixed lever arms, the mapping between the vector of muscle forces f and the joint torques τ can be expressed as

$$\tau = M \cdot f = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,m} \\ m_{2,1} & m_{2,2} & \dots & m_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{n,m} \end{bmatrix} \cdot f \tag{4.36}$$

while M denotes the $n \times m$ matrix of lever arms, whereas $m_{k,j}$ represents the lever arm of muscle j on the k^{th} component of the torque vector. If a muscle does not affect a joint, the respective lever arm evaluates to zero.

In cases where muscles do not collide with the skeleton (see also Fig. 4.5), they can be modeled as direct lines between the anchor points (see Fig. 4.4a) and the torque applied on a single joint by a muscle is expressed in three dimensional space by the general force-torque relationship [78]

$$\vec{\tau} = \vec{r} \times \vec{f} \tag{4.37}$$

where $\vec{\tau}$ denotes the three-dimensional torque vector. While Eq. 4.37 can be evaluated for arbitrary coordinate frames, the respective elements of the joint torque vector τ_i are formulated in the moving joint frame f'_{J_i} in this work. Assuming that $\vec{\tau}$ is expressed in the base frame f_0, the scalar joint torque of a revolute joint is defined as the scalar product of the three-dimensional torque vector and the unit vector along the joint axis

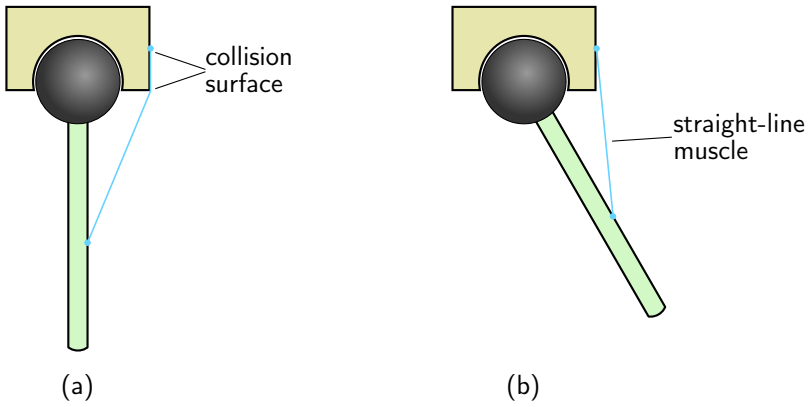


Fig. 4.5: Muscle Collisions. In the presence of spherical joints, tendon routing becomes difficult as it is not possible to apply pulleys. Therefore, collisions between muscles and the skeletal structure, depending on the pose, are bound to take place. A spherical joint with a single muscle is shown for (a) the colliding and (b) the non-colliding case.

e_i. As spherical joints are free to rotate in all three axis, this constraint is not necessary. However, the torque due to the muscle force has to be transformed into the coordinate frame of the joint.

$$\tau_{i,j} = {}^0\vec{e}_i^T \cdot {}^0\vec{\tau}_{i,j} = {}^0\vec{e}_i^T \cdot ({}^0\vec{r}_{i,j} \times {}^0\vec{f}_j) \qquad \text{if joint } i = \text{revolute} \tag{4.38}$$

$$\tau_{i,j} = {}^{i\prime}R_0 \cdot {}^0\vec{\tau}_{i,j} = {}^{i\prime}R_0 \cdot ({}^0\vec{r}_{i,j} \times {}^0\vec{f}_j) \qquad \text{if joint } i = \text{spherical} \tag{4.39}$$

For multi-articular muscles, the muscle force f acts upon each of the affected joints, following the same rules (see Fig. 4.4c).

Eq. 4.38 and Eq. 4.39 can be rewritten to express the lever arms of the muscles.

$$\tau_{i,j} = \left[{}^0\vec{e}_i^T \cdot \left({}^0\vec{r}_{i,j} \times \frac{{}^0\vec{f}_j}{f_j} \right) \right] \cdot f_j = m_{k,j} \cdot f_j \qquad \text{if joint } i = \text{revolute} \tag{4.40}$$

$$\tau_{i,j} = \left[{}^{i\prime}R_0 \cdot \left({}^0\vec{r}_{i,j} \times \frac{{}^0\vec{f}_j}{f_j} \right) \right] \cdot f_j = \begin{bmatrix} m_{k+0,j} \\ m_{k+1,j} \\ m_{k+2,j} \end{bmatrix} \cdot f_j \qquad \text{if joint } i = \text{spherical} \tag{4.41}$$

However, the lever arms now depend on the joint configuration. Therefore, the matrix of lever arms becomes a function of the joint angles q.

$$\tau = M(q) \cdot f \tag{4.42}$$

The lever arm approach is feasible for tendon-driven robots, even in the presence of more complex joint types and bi-articular muscles. It has been used e.g. to describe the muscle kinematics in the BioBiped [107]. However, in musculoskeletal robots, the assumption of

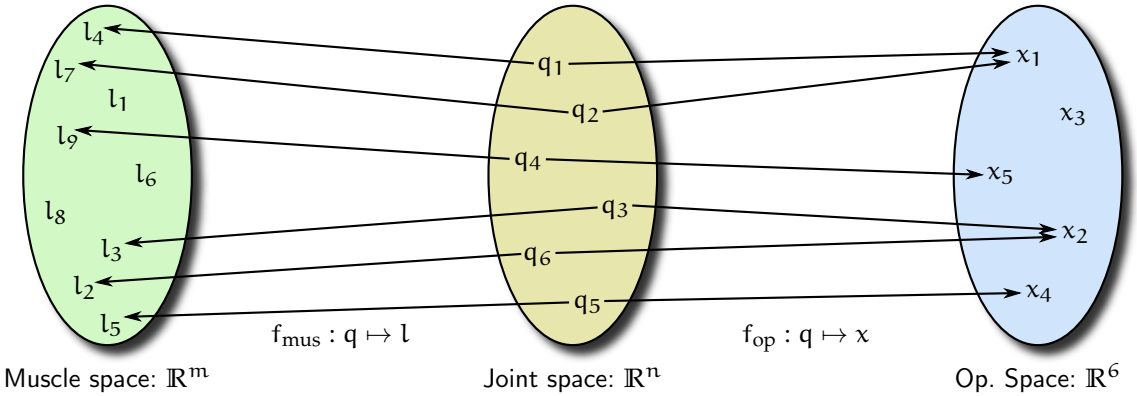


Fig. 4.6: Relating Joint, Operational and Muscle Space. Joint angles q can be mapped onto operational space positions x, defined by the mapping f_{op}, as well as the muscle lengths l, defined by the mapping f_{mus}.

straight-line muscles does not hold, due to collisions between muscles and the skeletal structure and even collisions in between muscles (see Fig. 4.5). In this case, a different route has to be taken.

From Section 4.1.1 we know that there is a duality in the instantaneous kinematics, describing the relationship between operational space force and joint torques. The joint space is defined as the space of possible joint angles q which is in $\mathbb{R}^n$, with n as the number of skeletal DoF. There exists a mapping f_{op} from joint angles to operational space coordinates which is in $\mathbb{R}^6$, considering three positional and three rotational DoF (see Fig. 4.6).

$$f_{op}: \quad \mathbb{R}^n \to \mathbb{R}^6 \qquad q \mapsto x(q) \tag{4.43}$$

This mapping is neither *injective* nor *surjective*. A similar mapping f_{mus} exists for mapping the joint angles q into the muscle space which is in $\mathbb{R}^m$, with m being the number of muscles. Therefore, f_{mus} can be defined as follows.

$$f_{mus}: \quad \mathbb{R}^n \to \mathbb{R}^m \qquad q \mapsto l(q) \tag{4.44}$$

In the absence of friction, the muscle length is the length of the shortest path between the two muscle anchor points which does not collide with any other part of the robot. It is affected by the coordinates of the joints that are spanned by the respective muscles. Note that this description can be extended towards muscles with multiple anchor points by adding the lengths of the segments between the anchor points. In this case f_{mus} is *injective* for $m > n$, but not *surjective*. Therefore, neither f_{mus} nor f_{op} is invertible, but due to the fact that f_{mus} is *injective* it is possible to observe the joint angles through measurements of l. Muscles that span the same joint are dependent on each other. By capturing these dependencies in constraints on the muscle lengths, the muscle space can be constrained to valid vectors of l, rendering f_{mus} to be *surjective* as well.

Note that f_{mus} is always differentiable in the case of straight line muscles. In contrast, muscles that wrap around skeletal structures, might be affected by hysteresis effects due to friction. This sliding between tendon and skeleton, renders the mapping from joint to muscle space to be not unique. Even worse, muscles might flip about protruding structures and the mapping becomes not differentiable. Therefore, care has to be taken that the path of the tendons is clearly defined at all times when the robot is designed. Throughout this work, differentiability and uniqueness of f_{mus} are assumed to be ensured by the robot design.

Differentiating f_{op} with respect to the joint angles, the relative kinematics or joint Jacobian is obtained (see Section 4.1.1). Similarly, the muscle Jacobian $L(q)$ can be obtained by differentiating f_{mus} with respect to the joint angles [149]. It gives a relation between the time derivative of the muscle lengths l and the joint velocities $\dot{q}$ for a given pose.

$$L(q) = \frac{\partial l}{\partial q} \tag{4.45}$$

$$\frac{\partial l}{\partial t} = L(q) \cdot \dot{q} \tag{4.46}$$

Similar to the joint Jacobian (see Section 4.1.1), the *principle of virtual work* can be applied [22], stating equality of the instantaneous amount of work applied in both spaces. The instantaneous work performed on the system is equal to the muscle force times the linear coordinate of the kite line. This work is split into the work performed in the joint space plus the work performed on the elastic elements (see Fig. 4.3).

$$f^T \cdot \partial x' = \tau^T \cdot \partial q + f^T(\partial x' - \partial x) \tag{4.47}$$

The instantaneous change in length can be obtained from Eq. 4.30,

$$l = l_o - R_S\theta + K^{-1}f = l_0 - x \tag{4.48}$$

$$\partial l = -\partial x \tag{4.49}$$

leading to.

$$-f^T \cdot \partial l = \tau^T \cdot \partial q \tag{4.50}$$

Expanding by $\frac{1}{\partial t}$, and inserting Eq. 4.46 we obtain

$$-f^T \cdot L(q) \cdot \frac{\partial q}{\partial t} = \tau^T \cdot \frac{\partial q}{\partial t} \tag{4.51}$$

which can be simplified to yield the following relation between muscle forces and joint torques.

$$\tau = -L^T(q) \cdot f \tag{4.52}$$

This dual description, for the force to torque and the joint velocity to muscle velocity relationship is extremely useful. It has been utilized in the simulation of biomechanical

structures to solve the problem of obtaining the muscle lever arms [22]. By finding the mapping f_{mus} and drawing the partial derivative, this problem can be solved in an elegant way. Therefore, from Eq. 4.42 and Eq. 4.52 it can be concluded that the muscle Jacobian describes the matrix of lever arms as follows.

$$-L^T(q) = M(q) \tag{4.53}$$

A general solution for f can be formulated, utilizing the pseudo inverse L^{+T} of the muscle Jacobian transpose [2],

$$f = -L^{T+}(q) \cdot \tau + L_0(q) \cdot f_0 \qquad f_0 \in \mathbb{R}^{(m-n)} \tag{4.54}$$

where f_0 is an arbitrary vector and $L_0(q)$ denotes the $m \times (m-n)$ null-space matrix of L^T such that

$$-L^T(q) \cdot L_0(q) = 0 \tag{4.55}$$

While the formulation of the null-space is useful to account for the effects of co-contraction, the general solution due to the pseudo inverse is in itself not usable for control, as it does not consider the fact that muscle forces always have to be positive, i. e. muscles can only pull not push.

In the presence of spherical joints, the pose of the robot cannot be represented by angles, leading to the choice of utilizing quaternions in the pose vector α (see Section 4.1). In this case the relation of the derivative of α and the joint velocities $\dot{q}$ affects the calculation of the muscle Jacobian. By defining $L_\alpha(\alpha)$ as the partial derivative of the muscle lengths with respect to the pose vector as follows, the muscle Jacobian L can be derived.

$$L_\alpha(\alpha) = \frac{\partial l}{\partial \alpha} \tag{4.56}$$

After solving Eq. 4.56 for ∂l and dividing by ∂t, the relation between the derivative of the positional coordinate vector and the joint velocities $A(\alpha)$ from Eq. 4.16 can be introduced, yielding

$$\frac{\partial l}{\partial t} = L_\alpha(\alpha)\frac{\partial \alpha}{\partial t} = L_\alpha(\alpha) \cdot A(\alpha)\frac{\partial q}{\partial t} \tag{4.57}$$

and the muscle Jacobian $L(\alpha)$ can be rewritten as follows.

$$L(\alpha) = \frac{\partial l}{\partial q} = L_\alpha(\alpha) \cdot A(\alpha) \tag{4.58}$$

Hence a useful tool has been created to capture the muscle kinematics, which relates the muscle with the joint space and can be applied even in the case of colliding muscles and complex joints. The difficulty, however, lies in obtaining the mapping between the joint and the muscle space f_{mus}. If this mapping can be established and it is differentiable, partial differentiation can be applied to gain the muscle Jacobian.

4.3.2 *Polynomial Regression Approximation*

The muscle Jacobian can be obtained in different ways. One possibility is to find a geometric representation of the muscle lengths subject to the joint coordinates and subsequently differentiating with respect to the joint angles. Finding this geometric representation is straightforward for a single direct line muscle, as the muscle length $l(q)$ can be expressed as follows

$$l(q) = \| {}^{a}x_1 - {}^{a}T_b(q) \cdot {}^{b}x_2 \| \tag{4.59}$$

where ${}^{a}x_1$ is the position of the first anchor point in the coordinate frame of link a and ${}^{b}x_2$ is the position of the second anchor point in the coordinate frame of link b. The transformation matrix ${}^{a}T_b$ gives the positional and rotational transform between the two bones and is a function of the joint coordinates that are spanned by the muscle. In case of a uni-articular muscle, this is simply the rotation of the joint. In case of a multi-articular muscle, ${}^{a}T_b$ has to combine the transform of all joints and rigid links in between the muscle anchor points.

This method can be extended towards colliding muscles, if the shapes of the colliding objects are clearly defined. However, for robots with many muscles and possibly also more complex joints this task becomes extremely tedious. Therefore, another possibility is proposed which is to numerically approximate the joint angle to muscle length mapping in Eq. 4.44 by drawing samples of muscle lengths and corresponding joint angles. Subsequently a regression problem can be formulated to obtain the unavailable mapping. Samples can be obtained by utilizing a force control algorithm to maintain a given tension and manually moving the joint, covering the work space several times. By recording motor positions, joint angles and muscle forces, different means of numerical approximations can be used to obtain Eq. 4.44. If only samples within a certain force boundary are taken into account, the expansion of the elastic element can be neglected. Therefore, there is no need to measure the muscle expansion (or to utilize a model for the elastic element to compute it from the force). In this case the muscle lengths l can be inferred from the motor positions θ, utilizing the spindle radius R_S.

$$l = l_0 - R_S \cdot \theta \tag{4.60}$$

It is important to note that friction in the force transmission of the muscles, due to deflection mechanisms like pulleys or eyelets (see Chapter 3), can affect the quality of the samples. In the case where the muscle force is measured in the motor unit, like it is the case for the Anthrob, the force acting on the limbs is distorted by friction. Hence, the force measurement deviates from the muscle force which is acting on the elastic element. The friction losses in the transmission system depend mainly on the transmitted force, as well as the direction of the movement. Therefore, the error due to the friction losses can be reduced to a minimum by choosing the control force to be relatively small, leading to friction losses to be very low, as well. Furthermore, the direction dependency is symmetric around the operating point which can be addressed by drawing samples coming from

all directions. Regression based on the minimum square error will place the approximation to be in the center between the different samples leading to a result that is close to the actual value.

In the case where the relationship between joint angle and muscle lengths is one dimension, i. e. each muscle length depends only on a single DoF, the muscle length function can be approximated, using polynomial regression. This is the case for all uni-articular muscles that span only revolute joints. Hence, the fitting problem for each muscle can the be formulated as

$$\begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_o \end{bmatrix} = \begin{bmatrix} 1 & q_1 & q_1^2 & \dots & q_1^d \\ 1 & q_2 & q_2^2 & \dots & q_2^d \\ 1 & q_3 & q_3^2 & \dots & q_3^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & q_o & q_o^2 & \dots & q_o^d \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} \tag{4.61}$$

where d is the degree of the polynomial, used for the approximation and o is the number of samples (see Section 2.3.1). The muscle Jacobian entry for each muscle L_i can subsequently be approximated as follows.

$$L_i = \begin{bmatrix} 1 \\ 2 \cdot q \\ \vdots \\ d \cdot q^{d-1} \end{bmatrix}^T \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} \tag{4.62}$$

If only the muscle Jacobian and not the approximation of the muscle length is of interest, it is not necessary to get a valid initial muscle length l_0 as it vanishes during differentiation.

As an example, the muscle Jacobian was approximated for the elbow joint of the Anthrob with the two uni-articular muscle, i. e. the Brachialis and the Triceps. For this purpose, samples of joint angles and corresponding motor positions and forces were gathered at a sampling frequency of 50 Hz, while muscle force control was utilized to maintain a minimum force of 1 N. The operational range of the elbow was covered four times. After removing all samples with a muscle force that was outside of a ± 0.2 N range, the number of samples was reduced to 1708. Good fits for the muscle lengths were achieved (see Fig. 4.7) with only a 2nd order polynomial for the Triceps (coefficient of determination: $R^2 = 0.99999$) and a 3rd order polynomial for the Brachialis ($R^2 = 0.99991$).

As a comparison, a geometric solution, utilizing the CAD design of the robot, was determined under the assumption of straight line muscles. Here, Eq. 4.59 can be simplified due to uni-articular muscles with a single revolute joint.

$$l(q) = \| {}^a x_1 - R_Z(q) \cdot {}^b x_2 \| \tag{4.63}$$

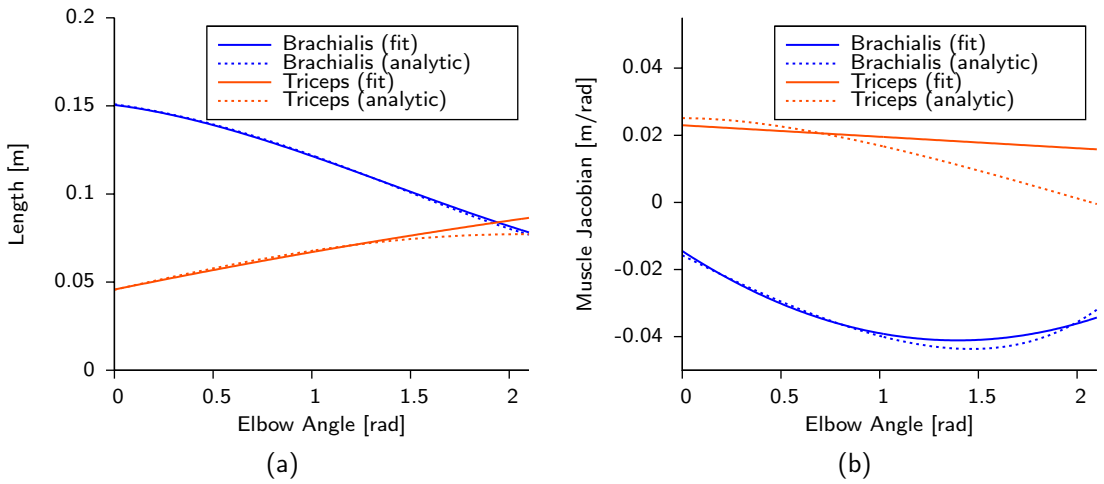


Fig. 4.7: Approximation of the muscle kinematics for the Anthrob elbow joint. In (a) the muscle lengths of both polynomial regression and the geometric solution are shown, and (b) depicts the resulting muscle Jacobian.

with R_Z being the rotation matrix around the rotation axis (z) of the elbow. Fig. 4.7a depicts a comparison of the geometric solution and the approximation. This led to a good representation for the muscle length of the Brachialis, even though the anchor points cannot be uniquely determined due to the movement of the kite line inside the guiding eyelets. The ceramic guidances allow for a movement of the attachment point inside the plane of the eyelet which has a radius of 3.5 mm. For the Triceps, however, the deviations became significant. It can be seen that the Triceps cannot be modeled as a straight line muscle, as it collides both with the lower as well as the upper arm, depending on the posture.

Fig. 4.7b shows again a comparison between the geometric solution for the muscle Jacobian and the polynomial fit. While deviations are comparably small for the Brachialis—with respect to measurement accuracy and the fact that anchor point locations change with the posture—the Triceps deviations are again significant. Interestingly, it can be seen from the data that the Triceps could be modeled by assuming a constant muscle Jacobian of ~ 0.02 m.

The presented results clearly show that muscle collisions excessively influence the muscle kinematics. These effects can only be modeled from the collision shapes, available from CAD data, if measures for guiding the tendons, e. g. pulleys, are introduced. However, when more complex assemblies like spherical joints or multi-articular muscles are considered, this is not always possible.

4.3.3 *Neural Network Approximation*

For cases where the muscle lengths depend on multiple DoF, the approach of polynomial regression is not suited and different approximation methods should be used. One possible method is the utilization of Artificial Neural Networks (ANNs) as a regressor for the multi-dimensional mapping Eq. 4.44 (see Section 2.3.2). The approximation of the muscle Jacobian through ANN has been extensively studied by Schmaler [114] for the simulation of a musculoskeletal robot arm. As samples can be drawn from the full joint space, the function approximator needs only to interpolate between samples, without the need for extrapolation. Therefore, a Multilayer Perceptron (MLP) network with a single hidden layer was chosen.

Similar to the approximation by polynomial regression, samples of joint angles and corresponding muscle lengths are generated, prior to network training. Subsequently, the ANN is presented *offline* with the samples, posing a supervised learning problem which can be treated using the well-known back-propagation algorithm (see Section 2.3.2.2 and [137]). In general, the regression can be divided into subproblems, since not all muscles are related to all joints. To create simpler learning subtasks and to relieve the learning algorithm of having to identify the structure of the robot, muscles that span specific joints are represented by a network that takes only the corresponding joint angles as inputs, while bi-articular muscles will depend on joint angles from multiple joints and need to be represented by a separate network.

Subsequently the ANN approximation is differentiated with respect to the joint angles, using the difference quotient to obtain the muscle Jacobian. In the scalar case this can be written as follows,

$$L(q) = \frac{f_{mus}(q + 0.5 \cdot \Delta) - f_{mus}(q - 0.5 \cdot \Delta)}{\Delta} \tag{4.64}$$

where Δ is the finite difference chosen for the approximation. For multi-dimensional pose vectors α, the difference quotient can be easily extended,

$$L_\alpha(\alpha)_{ij} = \frac{f_{mus,i}(\alpha + 0.5 \cdot \Delta \cdot e_j) - f_{mus,i}(\alpha - 0.5 \cdot \Delta \cdot e_j)}{\Delta} \tag{4.65}$$

while $f_{mus,i}$ is the i^{th} entry of the muscle length approximation and e_j denotes the unit vector of the dimensionality of α for the j^{th} pose vector entry.

In [114] the model of a musculoskeletal arm was simulated within CALIPER, a generic simulation platform based on the open source physics engine Bullet [18]. This platform was specifically developed for the simulation of musculoskeletal robots [141]. The simulation facilitated the extraction of samples, as it gave the possibility to drive the joints directly into arbitrary positions and extract the corresponding muscle lengths. It was hence possible to gather samples from the complete operational range of the arm. Furthermore ground-truth data for the muscle Jacobian could be generated by driving the joints to a given pose and subsequently extract the gradient of the muscle lengths at that

Table 4.1: ANN Approximation for a Simulated Musculoskeletal Arm. Both the normalized mean square error, as well as the mean absolute error is given for the learned muscle length function and the resulting muscle Jacobian. [114]

muscle	neurons	muscle length		muscle Jacobian	
		nMSE	$\overline{\text{error}}$ [m]	nMSE	$\overline{\text{error}}$ [m]
Biceps	20	$3.76 \cdot 10^{-5}$	$1.57 \cdot 10^{-4}$	$2.44 \cdot 10^{-1}$	$3.33 \cdot 10^{-3}$
Brachialis		$1.41 \cdot 10^{-5}$	$5.96 \cdot 10^{-5}$	$7.90 \cdot 10^{-1}$	$1.06 \cdot 10^{-4}$
Triceps	5	$2.44 \cdot 10^{-6}$	$5.26 \cdot 10^{-5}$	$8.46 \cdot 10^{-1}$	$1.06 \cdot 10^{-4}$
Ant. Deltoid		$9.30 \cdot 10^{-5}$	$1.73 \cdot 10^{-4}$	$6.18 \cdot 10^{-1}$	$3.63 \cdot 10^{-3}$
Infra		$5.07 \cdot 10^{-5}$	$6.18 \cdot 10^{-5}$	$5.05 \cdot 10^{-1}$	$4.40 \cdot 10^{-3}$
Lat. Deltoid		$9.03 \cdot 10^{-5}$	$1.51 \cdot 10^{-4}$	$5.72 \cdot 10^{-1}$	$3.76 \cdot 10^{-3}$
Pectoralis		$5.75 \cdot 10^{-5}$	$1.08 \cdot 10^{-4}$	$5.39 \cdot 10^{-1}$	$4.53 \cdot 10^{-3}$
Post. Deltoid	30	$1.02 \cdot 10^{-4}$	$1.31 \cdot 10^{-4}$	$5.26 \cdot 10^{-1}$	$3.84 \cdot 10^{-3}$
Supra		$7.96 \cdot 10^{-5}$	$6.76 \cdot 10^{-5}$	$4.84 \cdot 10^{-1}$	$3.65 \cdot 10^{-3}$
Teres Major		$5.44 \cdot 10^{-5}$	$1.07 \cdot 10^{-4}$	$4.83 \cdot 10^{-1}$	$3.99 \cdot 10^{-3}$
Teres Minor		$3.13 \cdot 10^{-5}$	$5.30 \cdot 10^{-5}$	$5.01 \cdot 10^{-1}$	$3.95 \cdot 10^{-3}$
Average		$5.57 \cdot 10^{-5}$	$1.02 \cdot 10^{-4}$	$5.55 \cdot 10^{-1}$	$3.21 \cdot 10^{-3}$

position, by again driving the joint according to Eq. 4.65. This way it was possible to evaluate different ANN configurations against the muscle Jacobian that was extracted directly from the simulation. The simulation model used here is qualitatively equal to the Anthrob, containing a spherical shoulder joint and a revolute elbow joint. Actuation was realized by eleven muscles of which one was bi-articular. However, it has to be noted that dimensions and parameters of the simulation model were different from the Anthrob.

Altogether a total of 103 055 samples were generated. The ANNs were structured, such that the one bi-articular muscle in the robot was represented by a separate neural network, taking the quaternion of the shoulder joint and the angle of the elbow as an input. The muscles on the elbow joint, i. e. the Triceps and Brachialis, were incorporated in a second network. Finally, the remainder of the muscles that only affect the shoulder were covered by a third network. The network sizes were determined experimentally and were chosen as shown in Table 4.1.

While the average error in the length estimation is as low as 0.102 mm, the average error in the muscle Jacobian is already 3.21 mm (see Table 4.1). This is still within reasonable bounds for the application considered here. When this method is applied to learning the muscle Jacobian for a physical robot, sensor noise and measurement errors can increase the error in the muscle Jacobian. Hence, special attention has to be paid to the validation error, since larger errors can degrade performance of control laws that are based on the muscle Jacobian.

Table 4.2: ANN Approximation for the Muscle Lengths in the Anthrob. For comparison with Table 4.1, the mean validation error is given, together with the MSE and the RMSE.

muscle	neurons	samples	$\overline{\text{error}}$ [m]	MSE [m^2]	RMSE [m]
Biceps	20	76 255	$2.16 \cdot 10^{-4}$	$8.14 \cdot 10^{-8}$	$2.85 \cdot 10^{-4}$
Brachialis			$1.90 \cdot 10^{-4}$	$6.59 \cdot 10^{-8}$	$2.57 \cdot 10^{-4}$
Triceps	5	1708	$9.96 \cdot 10^{-5}$	$2.38 \cdot 10^{-8}$	$1.54 \cdot 10^{-4}$
Ant. Deltoid			$2.96 \cdot 10^{-4}$	$1.70 \cdot 10^{-7}$	$4.12 \cdot 10^{-4}$
Infra			$2.59 \cdot 10^{-4}$	$1.55 \cdot 10^{-7}$	$3.94 \cdot 10^{-4}$
Lat. Deltoid			$2.99 \cdot 10^{-4}$	$1.60 \cdot 10^{-7}$	$4.00 \cdot 10^{-4}$
Pectoralis			$1.86 \cdot 10^{-4}$	$8.02 \cdot 10^{-8}$	$2.83 \cdot 10^{-4}$
Post. Deltoid	30	42 208	$2.56 \cdot 10^{-4}$	$1.05 \cdot 10^{-7}$	$3.24 \cdot 10^{-4}$
Supra			$2.24 \cdot 10^{-4}$	$8.34 \cdot 10^{-8}$	$2.89 \cdot 10^{-4}$
Teres Major			$2.16 \cdot 10^{-4}$	$7.46 \cdot 10^{-8}$	$2.73 \cdot 10^{-4}$
Teres Minor			$2.89 \cdot 10^{-4}$	$1.54 \cdot 10^{-7}$	$3.92 \cdot 10^{-4}$
Average			$2.30 \cdot 10^{-4}$	$1.05 \cdot 10^{-7}$	$3.15 \cdot 10^{-4}$

The method was also evaluated on the spherical shoulder joint of the Anthrob and the eight muscles spanning it. Using the same method as for the elbow joint, samples of corresponding joint angles, motor positions, and forces were generated by setting the muscle force control to maintain a minimum force of 2 N and sampling at a frequency of 50 Hz. The difficulty here was to cover the full operational space of the robot. The threshold for removing samples was in this case set to ± 1 N which led to a total of 42 208 samples which were split into a training set of 33 767 samples and a validation set of 8441 samples and used to train a neural network with 30 hidden neurons. The average validation RMSE was determined to be 0.346 mm (see Table 4.2) which, considering the measurement errors and the comparably high force threshold, shows that the approximated muscle Jacobian is of sufficient accuracy. In contrast to the data, gathered from the simulation (see Table 4.1), no ground truth data was available for the muscle Jacobian and therefore no error measure could be computed directly. However, it can be seen that the mean validation errors for the muscle lengths are in the same order of magnitude as the mean error in simulation. It was shown in [114] that additive white noise had no measurable effect on the quality of the learning. Therefore, the good validation error leads to the conclusion that the learning of the muscle length mapping was successful and the extracted muscle Jacobian can be used in controlling the shoulder joint of the Anthrob. Similarly, the muscle Jacobian was obtained for the muscles of the elbow joint and the bi-articular Biceps (see Table 4.2).

In Fig. 4.8 the learned muscle Jacobian is shown for two muscles, the Lateral Deltoid and the Pectoralis, giving examples for the muscle kinematics of the Anthrob. It can be

seen that muscle lever arms change heavily with the joint angle, which also shows how muscles take on different roles, depending on the joint angle. E. g. the lever arm of the Pectoralis around the X-Axis changes by a factor of three, as the shoulder moves around the Y-Axis (see Fig. 4.8d).

4.4 MUSCULOSKELETAL ROBOT MODEL

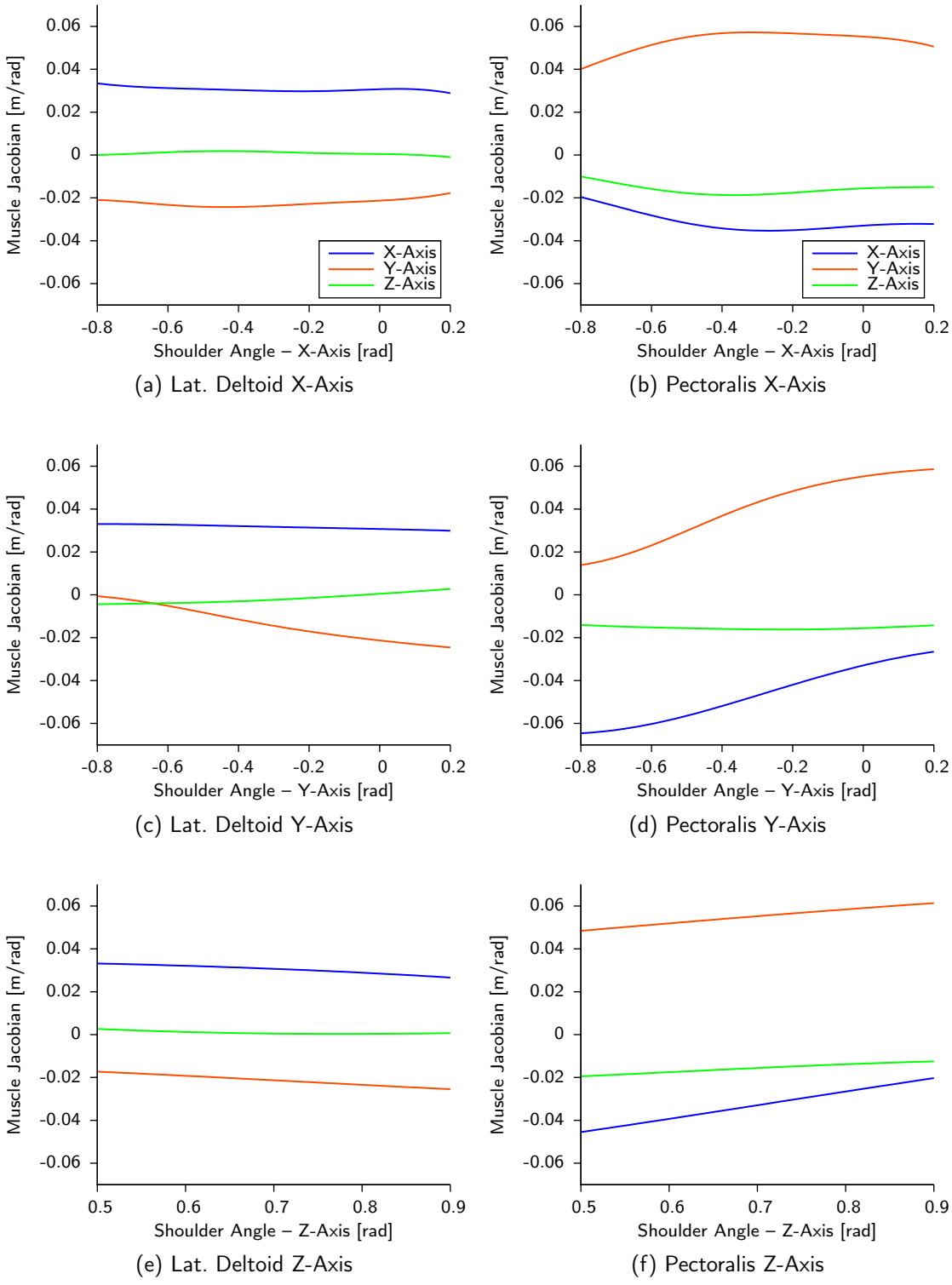
To obtain a model for the full robot from the submodels developed in the previous three Sections, the force to torque relationship in Eq. 4.52 provided by the muscle Jacobian can be introduced into the equation of motion for the skeleton Eq. 4.14.

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_G(q) = -L(q)^T \cdot f + \tau_{ext} \tag{4.66}$$

Subsequently, either the linear state space model in Eq. 4.33 or the non-linear relationship in Eq. 4.35 can be used to describe the muscle dynamics.

From the system at hand, several properties can be extracted. The fact that the mapping f_{mus} is *injective* makes the observation of the joint angles from measurements of the muscle lengths possible. From neural science we know that the human sensory motor system is equipped with muscle length and muscle tension sensors, i. e. the golgi tendon organs and the muscle spindles, respectively. It is believed that these are mainly used in motor control, while from the joint angle sensors, only poor quality signals can be obtained [56]. Therefore, by learning the mapping between joint and muscle space, it is possible to account for the missing joint angle sensors. However, for this purpose absolute and reliable muscle length measurements need to be available. By construction, this is not possible in the robot used in this work. The Anthrob features motor position and force sensors, which can only be used to approximate the muscle lengths if a highly accurate model of the elastic element is available. Due to visco-elastic effects this is not the case for the Nitrile Butein Rubber (NBR) rings, used here. This fact is compensated by the utilization of joint angle sensors, instead. These are also necessary to initially approximate the muscle to joint space mapping. In future robots, however, it would be possible to only obtain this mapping once, using e. g. a motion capture system. Subsequently the muscle kinematics can be used to compute joint angles from muscle lengths, hence rendering external sensing facilities or joint angle sensors unnecessary.

In joint actuated robots, the number of DoF is equal to the number of actuators, usually. This is generally not the case in musculoskeletal robots. Therefore, a distinction between the number of skeletal DoF n and the number of actuators m is made. Strictly speaking, the full number of DoF is therefore $n + m$, as the actuators and the joints are decoupled by the elastic element. Therefore, the class of musculoskeletal robots belongs to the class of underactuated systems. This means that not all DoF of the system can be controlled independently. However, if we constrain ourselves to the skeletal DoF, these robots can be controllable. A system is called *tendon controllable* if it is possible to obtain a valid force vector for any given vector of joint torques. For a force vector to be valid it has to be all positive, as tendons can only push, not pull. In other words, for any $\tau \in \mathbb{R}^n$, there exists



(a) Lat. Deltoid X-Axis (b) Pectoralis X-Axis

(c) Lat. Deltoid Y-Axis (d) Pectoralis Y-Axis

(e) Lat. Deltoid Z-Axis (f) Pectoralis Z-Axis

Fig. 4.8: Muscle Jacobian for the Anthrob Shoulder Joint. The ANN approximation is evaluated for two muscles, the Lateral Deltoid and the Pectoralis for three shoulder angle sweeps. One around each of the axis, while the Z-Axis zero position is at 0.7 rad.

$f \in \mathbb{R}^m_{\geqslant 0}$ that satisfies Eq. 4.52. Therefore, $L(\alpha)$ has to satisfy the following two conditions for the full joint space [2]:

1. $\text{rank}(L^T) = n$
2. The null-space matrix L_0 has an all positive column

While the first condition is trivial to obtain, the second one allows for obtaining an all positive force vector by choosing f_0, such that $f \in \mathbb{R}^m_{\geqslant 0}$ (see Eq. 4.54 and [64])

The additional control inputs, due to the higher number of actuators than the skeletal DoF, can be used to control the co-contraction of muscles. Generally, co-contraction is not necessary to achieve a certain movement. In this case, the internal forces should be reduced to a minimum. However, in cases where co-contraction is wanted, the null-space can be utilized by choosing a vector f_0 according to a task that is orthogonal to the movement of the joint.

5 FEEDBACK LINEARIZATION

The method of feedback linearization utilizes the non-linear model of a system to cancel all non-linear elements of the system dynamics (see Section 2.2.1). If the model describes all aspects of the system dynamics and the model parameters are perfectly known, a linear controller can be applied to stabilize the closed loop. For stiff robots this leads to the method of computed torque control [119]. Similarly, feedback linearization was used to develop controllers for the simulation of biomechanical structures, as well as for tendon-driven robots, leading to the technique called *computed muscle control* [130, 131]. Here a low-level muscle controller was utilized to first stabilize the muscle dynamics which was subsequently used by the high-level *computed muscle controller*. These previously presented control schemes assume perfect low-level control with respect to reference tracking. While this might be a valid assumption for some robots and respective controllers, the design of the muscle greatly influences the dynamics. In the following sections, different implementations of this family of control schemes are developed, discussing the drawbacks. To enable a comparison, several low-level controllers are developed (see Section 5.1). Subsequently, the method of *computed muscle control* is used to formulate three different control laws for joint space control.

In robot control we differentiate between (1) a regulation problem, where fixed coordinates are specified and the controller has the goal to bring and keep the robot to this joint configuration and (2) a trajectory tracking problem, where the robot is controlled to follow a trajectory, specified by positions, velocities and accelerations [119]. First, the different control laws are expressed as regulation problems in Section 5.2 and are subsequently extended towards trajectory tracking in Section 5.3.

5.1 LOW-LEVEL CONTROL

Due to the fact that the actuator dynamics for each of the muscles can be separated from each other, it is an obvious choice to find a low-level controller for each of the muscles that operates independently. This method has been used for flexible joint robots to control the joint torque [5], but also for tendon-driven robots. The independent control of the tendon forces has been proposed by Salisbury and Craig [113]. This approach has the advantage that low-level controllers can be implemented independently on distributed Electronic Control Units (ECUs), where they can possibly run at higher frequencies than the central controller. In contrast, Abdallah et. al. [1] formulate a control law based on a low-level actuator position controller. The advantage of the latter one is that the computation of the stiffness of the elasticity can be handled by the central controller, simplifying

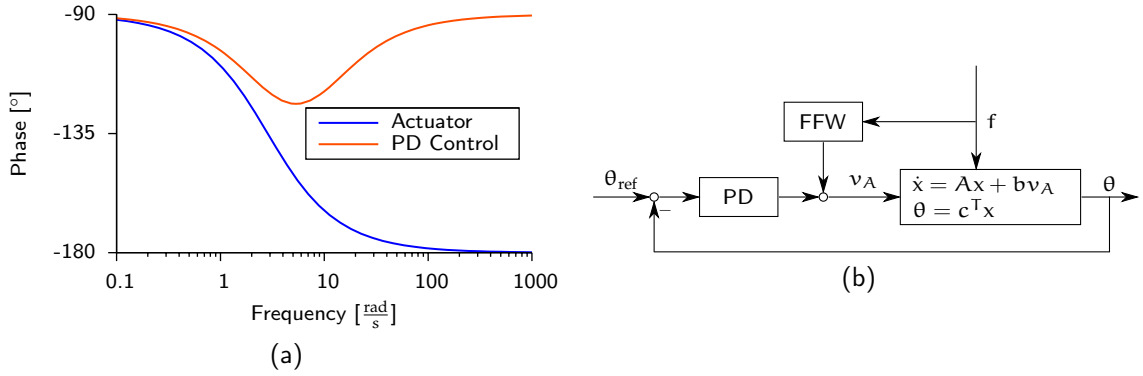


Fig. 5.1: Actuator Position Control. The (a) phase diagram for the actuator dynamics of the Anthrob shoulder muscles and for a PD control law is shown next to the (b) block diagram of a PD position controller with disturbance compensation.

the low-level control law. The first possibility, on the other hand, yields a better response to disturbances in the muscle forces, due to a possibly high control frequency of the low-level controller. Therefore, both a force and an actuator position controller are reformulated in the following sections.

5.1.1 *Actuator Position Control*

Controlling the position of an actuator based on a brushed DC motor, is a well known task. Standard techniques range from P controllers to a cascade of a current control loop and an outer PD motor position controller. Best results are obtained by realization of a fast underlying current control law [68]. The time constant for the motor current, can be identified as follows.

$$T_{\text{current}} = \frac{L_A}{R_A} \tag{5.1}$$

Depending on the time constant of the motor coil (see Section 4.2), control frequencies of 5 kHz or larger have to be realized.

There are cases, however, where these high control frequencies are not realizable with the available hardware. In these cases a PD controller which works directly on the actuator position (see Fig. 5.1b) can be used. Note that the closed loop dynamics are difficult to stabilize with an integrator in the control law. This becomes obvious when looking at the simplified motor dynamics

$$v_A = R_A i_A + \frac{c_M}{r_G}\dot{\theta} \tag{5.2}$$

$$\ddot{\theta} = \frac{c_M}{r_G J_T} i_A - \frac{R_S}{c_F r_G^2 J_T} f \tag{5.3}$$

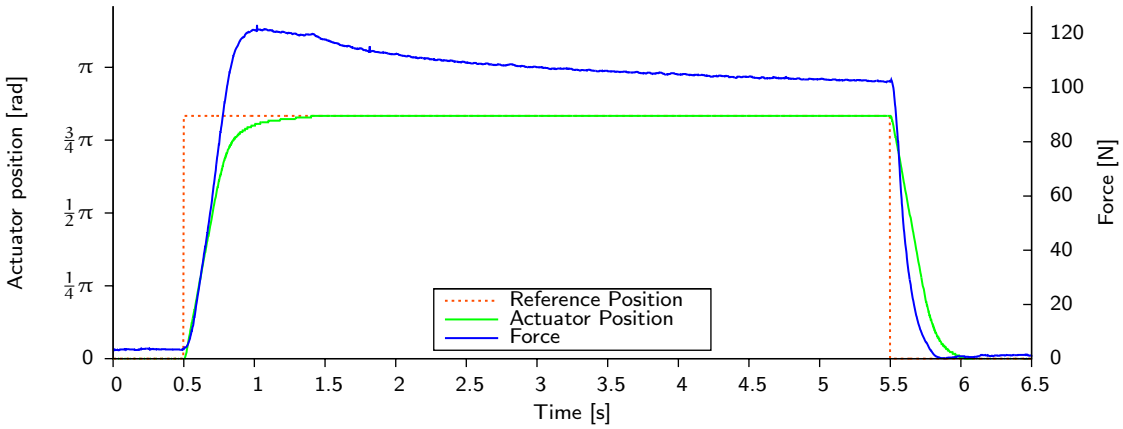


Fig. 5.2: Step Response for the Actuator Position Control. The actuator position is controlled to follow a step of 2.618 rad (150°). Both the actuator position, as well as the force is shown for a muscle of the Anthrob. The muscle length was kept constant throughout the experiment, by fixating the far end of the muscle.

that are obtained by taking the actuator dynamics without considering the effects of friction (see Section 4.2). A transfer function in the frequency domain for the actuator position can be obtained by setting the muscle force to zero.

$$\frac{\theta}{v_A} = \frac{\frac{c_M}{r_G J_T R_A}}{s^2 + \frac{c_M r_G}{R_A} \cdot s} \tag{5.4}$$

The phase diagram of this transfer function can be seen in Fig. 5.1a. It is obvious that an additional integrator in the controller will only enable a stabilization left of the phase drop, rendering the controller to be extremely slow. Therefore, an integrator is omitted. Instead a FFW term is introduced to reduce the steady state error (see Fig. 5.1b). This term can be computed by determining the steady state control input for a given force.

$$\mathrm{FFW} = v_A(0) = \frac{R_A R_S}{c_M c_F r_G} f \tag{5.5}$$

The resulting controller, as depicted in Fig. 5.1b, can be implemented without difficulties for the ECUs of the Anthrob with a control frequency of 1 kHz. While the choice of PD parameters is the main influence to the dynamics of the closed loop system, additional limitations are usually given by an output saturation (e. g. $\pm 12\,V$ in the case of the Anthrob). A step response for a typical muscle in the Anthrob is shown in Fig. 5.2. The rise time (10 % to 90 %) for the step response can be computed to be 0.31 s. Possibly, faster rise times are realizable by fine tuning the control parameters, however, the experiment shows the order of magnitude for a typical musculoskeletal robot muscle, which is mainly determined by the elastic element.

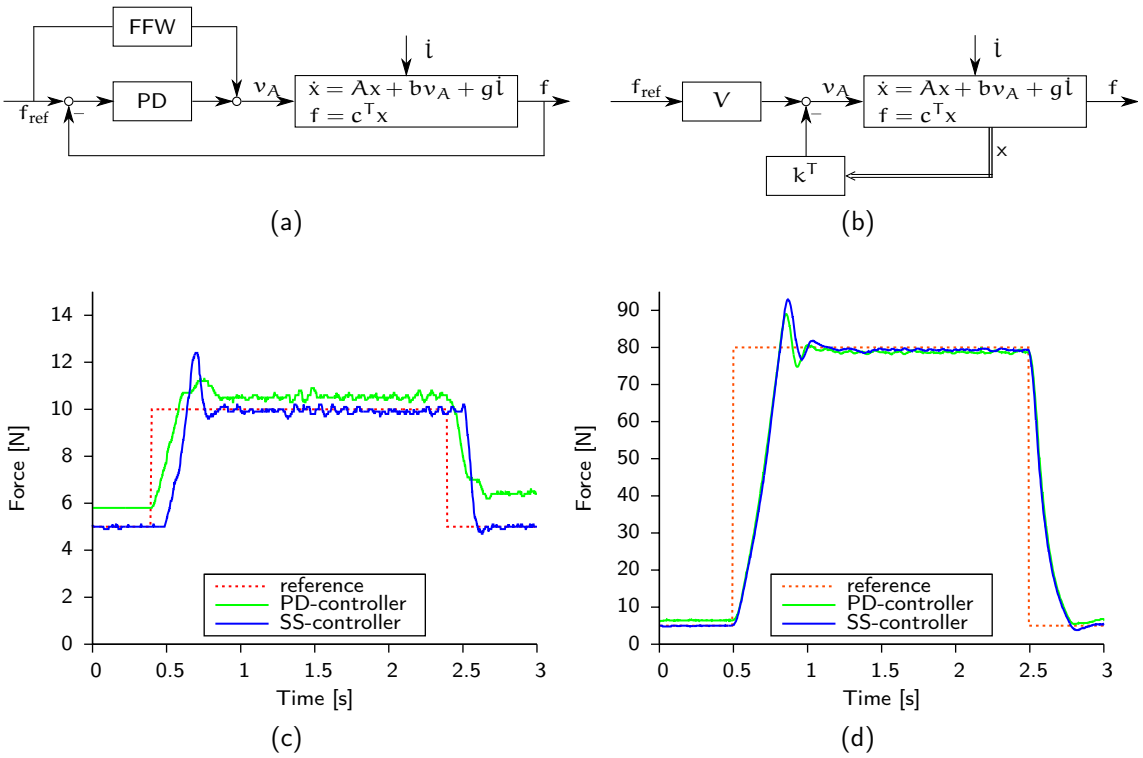


Fig. 5.3: Muscle Force Control. A comparison of a (a) PD force controller with a FFW term and a (b) state-space force controller k with a pre-filter V is shown for the linear system $\{A, b, c^T, g\}$ of the muscle model. The step response for a step from (c) 5 N to 10 N and (d) 5 N to 80 N shows the better performance of the state-space control scheme.

5.1.2 *Force Control*

Methods for muscle force control that have been previously proposed generally rely on a P or PD controller (see Fig. 5.3 and [50, 113]). This relatively simple control law, can be applied easily, as control parameters can be found by methods from linear control theory like direct design by root locus, utilizing the Nyquist stability criterion [28] or even by simple trial and error. Similar to the previously developed actuator position control a linear FFW term can be added. Due to the fact that the same effect is compensated, it computes exactly as stated in Eq. 5.5.

However, considering that the full state vector can be measured by the sensors available (see Section 3.5), it is also possible to develop a full state space controller. Reformulation

of the muscle dynamics model from Eq. 4.33 leads to a state space model of the following form,

$$\frac{d}{dt}\begin{bmatrix} i_A \\ \dot{\theta} \\ f \end{bmatrix} = \begin{bmatrix} -\frac{R_A}{L_A} & -\frac{c_M r_G}{L_A} & 0 \\ \frac{c_M}{r_G J_T} & -\frac{\mu_v}{c_F r_G J_T} & -\frac{R_S}{c_F r_G^2 J_T} \\ 0 & K \cdot R_S & 0 \end{bmatrix} \cdot \begin{bmatrix} i_A \\ \dot{\theta} \\ f \end{bmatrix} + \begin{bmatrix} \frac{1}{L_A} \\ 0 \\ 0 \end{bmatrix} \cdot v_A + \begin{bmatrix} 0 \\ 0 \\ K \end{bmatrix} \cdot \dot{l} \tag{5.6}$$

where the voltage v_A is taken as the control input and $\dot{l}$, the change in muscle length due to the movement of the skeleton, is handled as a disturbance. Therefore, Eq. 5.6 is rewritten, neglecting the disturbance.

$$\dot{x} = A \cdot x + b \cdot v_A \tag{5.7}$$

The advantage of such a control scheme is that the system dynamics can be fully taken into account and a controller can be developed by the method of closed loop pole shaping. The closed loop dynamics of the state space controller can be obtained by introducing the control law k^T into the system dynamics (see Fig. 5.3b).

$$\dot{x} = \left[A - I_3 \cdot bk^T\right] \cdot x \tag{5.8}$$

Solving this linear system of Ordinary Differential Equations (ODEs) leads to an exponential function with three poles. The method of shaping the closed loop system dynamics by placing the poles and subsequently computing the control gains k^T is known as Ackermann's formula [28]. In order for the closed loop system to be stable, the complex poles have to be all in the left half plane of the complex s-plane. Good control performance can be achieved by placing two poles to form a complex conjugate pair, exhibiting the desired damping behavior. By choosing the damping ratio ζ, a complex pole pair is computed as follows [28],

$$p_{1/2} = -\zeta \cdot \omega_n \pm \omega_n \cdot \sqrt{\zeta^2 - 1} \tag{5.9}$$

whereas ω_n defines how far left in the plane, the poles should lie, and therefore how fast the controller will converge. For the force controller the state vector is three dimensional and the third pole is chosen to be purely real and left of the complex pole pair.

This method can be applied in the continuous time domain, but also directly in the discrete time domain, taking the sampling time into account at design time. For this purpose, the state space muscle model is transferred into the discrete time domain, where the discrete time state space model is defined as follows.

$$x[k+1] = A_d x[k] + b_d \cdot v_A \tag{5.10}$$

The discrete parameters are defined with respect to the sample time T. There exist several methods of transferring continuous time models into discrete time models, starting from integration to zero-pole matching, and finally zero-order hold. The latter assumes that the continuous signal at the sampling instance is held until the next instance. This behavior is equivalent to what a discrete controller observes. For the zero-order hold method, A_d and b_d can be expressed as follows [28].

$$A_d = e^{A \cdot T} \tag{5.11}$$

$$b_d = A^{-1} \cdot (A_d - I_3) \cdot b \tag{5.12}$$

The method of pole shaping can be applied equally to the continuous and the discrete time model. First, the continuous poles p are chosen, as stated by Eq. 5.9, and subsequently transferred into the discrete time domain to obtain p_d.

$$p_d = e^{p \cdot T} \tag{5.13}$$

Subsequently, Ackermann's law can be applied in the discrete time domain, to obtain the digital control gains k_d^T. As a counterpart to the FFW term, a pre-filter V can be introduced for compensating the steady state error. The continuous time closed loop dynamics are rewritten to include the pre-filter (see Fig. 5.3b).

$$\dot{x} = \left[A - I_3 \cdot b k^T\right] \cdot x + b V \cdot f_{ref} \tag{5.14}$$

By setting the control error to zero as time goes to infinity, the pre-filter V is computed as follows [28].

$$\begin{bmatrix} M_x \\ M_u \end{bmatrix} = \begin{bmatrix} A & b \\ c^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{5.15}$$

$$V = M_u + k \cdot M_x \tag{5.16}$$

The standard control method of a PD controller with a FFW term was compared against the state space controller on a typical muscle of the Anthrob, performing a step up and step down from 2 N to 10 N and back and a second step to 80 N (see Fig. 5.3c and Fig. 5.3d). It can be seen that the state-space controller performs slightly better in tracking the reference than the PD controller. For this reason the state-space controller is utilized in the following for force control.

5.2 REGULATION

The well known method of computed torque control, utilizes the canonical form of the robot dynamics to linearize the system, by compensating the non-linear dynamics (see Section 2.2.1). Therefore, a torque is computed that according to the joint space system dynamics in Eq. 4.14 results in a given joint acceleration for the current system states,

defined by the pose vector and the joint velocities. In case of a perfect model, this method will cancel the system dynamics and the closed loop system will show the behavior defined by the linear control law. For regulation, a suitable linear control law is e. g. a PID controller, giving asymptotic stability and reducing the steady state offset, due to model uncertainties.

$$\nu = \ddot{q}_{ref} = P \cdot \Delta q + D \cdot \Delta \dot{q} + I \cdot \sum_{i=0}^{i=\frac{t}{\Delta t}} \Delta q_i \tag{5.17}$$

Depending on the application, a PD control law could also be sufficient. Obtaining Δq for revolute or linear joints is as simple as subtracting the two vectors q and q_{ref}. However, for quaternions Q and Q_{ref}, the corresponding ΔQ is not the difference, but the rotation between the two, which is defined as follows (see also Section A.2).

$$\Delta Q = Q_{ref} * Q^{-1} = \begin{bmatrix} \eta_{ref}\eta + \vec{\epsilon}_{ref}^{T}\vec{\epsilon} \\ \eta\vec{\epsilon}_{ref} - \eta_{ref}\vec{\epsilon} - S(\vec{\epsilon}_{ref})\vec{\epsilon} \end{bmatrix} \tag{5.18}$$

The zero rotation in quaternion notation is defined as $\Delta Q = [\pm 1 \quad 0 \quad 0 \quad 0]^T$. Therefore an error function Δq is chosen as follows [120] which is zero if and only if the rotation is zero

$$\Delta q = \eta\vec{\epsilon}_{ref} - \eta_{ref}\vec{\epsilon} - S(\vec{\epsilon}_{ref})\vec{\epsilon} \tag{5.19}$$

where $S(\cdot)$ represents the skew-symmetric matrix as stated in Eq. A.22. Therefore, the resulting Δq for a spherical joint is a three dimensional vector which points in the direction of the reference angle and is proportional to the offset. Convergence of the quaternion error dynamics can be proven by choosing a Lyapunov function candidate,

$$V = (\eta_{ref} - \eta)^2 + [\vec{\epsilon}_{ref} - \vec{\epsilon}]^T[\vec{\epsilon}_{ref} - \vec{\epsilon}] \tag{5.20}$$

which is zero if ΔQ is equal to $[1 \quad 0 \quad 0 \quad 0]^T$ and positive otherwise. The following closed loop dynamics, based on a P control law, are assumed for illustration.

$$\vec{\omega}' = P \cdot \Delta q \tag{5.21}$$

The relationship for mapping quaternion derivatives to rotational velocities are developed in Section A.2. Differentiating Eq. 5.20 and introducing the closed loop dynamics, as well as the relationship for mapping quaternion derivatives to rotational velocities (see Eq. A.37) leads to the following Lyapunov function derivative,

$$\begin{aligned} \dot{V} &= -2 \cdot (\eta_{ref} - \eta)\dot{\eta} + 2 \cdot [\vec{\epsilon}_{ref} - \vec{\epsilon}]^T(-\dot{\vec{\epsilon}}) \\ &= [\vec{\epsilon}\eta_{ref} - \eta\vec{\epsilon}_{ref} - S(\vec{\epsilon})\vec{\epsilon}_{ref}]^T P \Delta q \\ &= -\Delta q^T P \Delta q \end{aligned} \tag{5.22}$$

which is negative for positive definite proportional gains P unless Δq is zero. The duality of this rotation representation presents a problem in quaternion control. There exist unstable equilibrium points, e. g. $Q_{ref} = [1 \quad 0 \quad 0 \quad 0]^T$ and $Q = [-1 \quad 0 \quad 0 \quad 0]^T$, both denoting the zero rotation. In this configuration the Lyapunov function $V > 0$ and $\dot{V} = 0$. There are several approaches which solve this so called problem of unwinding for underwater or aerial vehicles [27, 77]. Practically the simplest solution is to constrain the valid quaternions, such that only half of the four dimensional unit sphere is used (see Section A.2). This can be easily realized on construction of the quaternion from the measurement of the angle and the reference quaternion from the reference angles, hence avoiding the unstable equilibria.

It can be seen that similar convergence proofs exist for other closed loop dynamics. Hence, the error in the joint position can be stated for the general case of a kinematic chain with revolute and spherical joints as follows.

$$\Delta q = \begin{bmatrix} \Delta q_1 \\ \vdots \\ \Delta q_N \end{bmatrix} \qquad \Delta q_i = \begin{cases} q_{iref} - q_i & \text{if joint } i = \text{revolute} \\ \eta_i \vec{\epsilon}_{iref} - \eta_{iref} \vec{\epsilon}_i - S(\vec{\epsilon}_{iref}) \vec{\epsilon}_i & \text{if joint } i = \text{spherical} \end{cases} \tag{5.23}$$

In the following, three control schemes are shown. The simplest, i. e. computed motor velocity, which neglects the system dynamics, utilizes solely the muscle kinematics, and operates on a low-level motor position control. Subsequently, Computed Force Control (CFC) and computed motor position control are developed which both use the full dynamic model of the skeleton and the muscle kinematics, while they operate on a low-level force and motor position control, respectively.

5.2.1 *Computed Motor Velocity Control*

For non-compliant robots, controllers are still formulated independently for each joint in some cases, rendering the overall control scheme to be only dependent on the robot kinematics. Transferring this to tendon-driven robots, leads to a controller which controls each actuator position, based on the joint movement. The definition of the muscle Jacobian (see Eq. 4.46) allows for computation of an actuator velocity for a given joint velocity if the spring expansion is neglected. Rewriting Eq. 4.30 and differentiating leads to a relationship of the change in muscle lengths and the actuator velocity $\dot{\theta}$

$$\dot{l} = \dot{\theta} \cdot r + \dot{f} \cdot K \tag{5.24}$$

which can be introduced into Eq. 4.46.

$$L(\alpha) \cdot \dot{q} = \dot{\theta} \cdot R_S + \dot{f} \cdot K \tag{5.25}$$

The low-level actuator position controller can be utilized to control each actuator, by integrating the actuator velocity. Here it is assumed that the position controller is fast

enough to reach the reference position by the time the next reference is calculated by the computed motor velocity controller for a control frequency T.

$$\dot{\theta} \cdot T = \theta_{ref} - \theta \tag{5.26}$$

This leads to the following control law, where the term $\dot{f} \cdot K$ is neglected.

$$\theta_{ref} = \dot{\theta} \cdot T + \theta = \frac{1}{r} L(\alpha) \cdot \dot{q} \cdot T + \theta \tag{5.27}$$

Obviously this suffers from several drawbacks. By neglecting the system dynamics it is impossible to account for the change in the spring expansion. In some robots this might be sufficient, however, in the case of a robotic arm these effects cannot be neglected. Furthermore, possible errors in the muscle Jacobian can lead to either slackening of the antagonistic muscles, or to a co-contraction, both of which are unwanted, unless being directly controllable.

5.2.2 *Computed Force Control*

If the dynamic effects of the robot are to be taken into account, a different route has to be taken. Here, computed torque control is extended to compute muscle forces from the dynamic model, hence accounting for the modeled dynamic effects. Solving the equation of motion for musculoskeletal robots (see Eq. 4.66) for the muscle forces f is underdetermined, as there are more muscles than DoF in the joints. This allows for adding additional constraints when solving for the muscle forces, by formulating an optimization problem [22], where different objective functions can be used. The most common possibility here, is to minimize the internal forces by minimizing the Euclidean norm [2, 22, 131]. However, it is also possible to e. g. optimize for energy consumption. There are two constraints. First, the forces are to apply a certain reference joint torque, and second, the muscle forces have a lower bound. By minimizing the muscle forces, using these two constraints, it is guaranteed that (1) the resulting forces lead to the desired behavior and (2) the internal forces of the system are kept at a minimum.

$$\min_{f_{ref}} \|f_{ref}\|^2 \tag{5.28}$$

$$\text{subject to} \begin{cases} -L^T(\alpha) \cdot f_{ref} = \tau_{ref} \\ f_{ref} \geqslant f_{min} \end{cases}$$

This well known optimization problem has the general form of a quadratic program,

$$\min_{x} \quad a^T x + \frac{1}{2} x^T G x \tag{5.29}$$

$$\text{subject to} \begin{cases} C_e x - c_e = 0 \\ C_i x - c_i \geqslant 0 \end{cases}$$

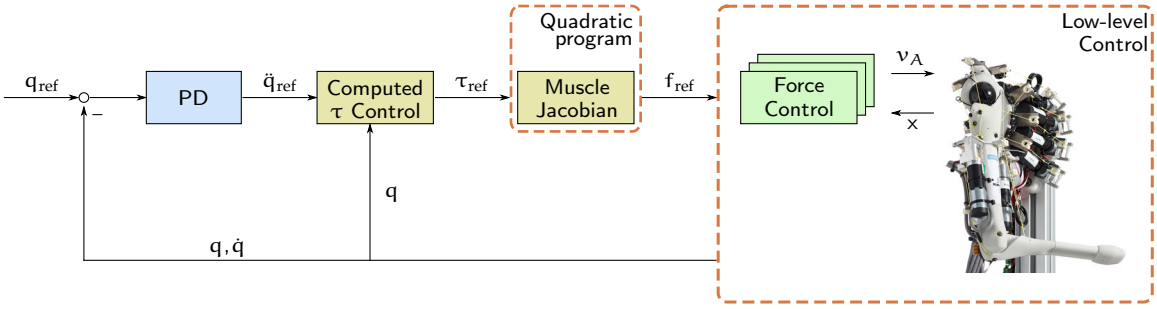


Fig. 5.4: CFC Block Diagram. A layout of the feedback linearization controller is shown. A reference acceleration is computed from a linear control law. Subsequently, the equation of motion is utilized to compute a reference torque, and finally a reference force is obtained from the quadratic optimization procedure.

while equality and inequality constraints are expressed by the sets $\{C_e, c_e\}$ and $\{C_i, c_i\}$, respectively. The solution to a quadratic program can be found by a large variety of optimization procedures, e. g. Wolfe [144] or Dantzig [20] which are extensions to the simplex method for linear programs. Here, better performance in the sense of computational time can be achieved by the method of Goldfarb and Idnani [34]. It uses the unconstrained minimum of the objective function as a starting point, subsequently adding the constraints.

In this case, where the Euclidean norm of the muscle forces is minimized, G evaluates to $diag(\frac{1}{2})$, which is positive definite. Therefore, the optimization problem is convex, leading to the fact that no local minima exist and convergence of the optimization procedure is guaranteed. Therefore, the implementation of the Goldfarb and Idnani method, used throughout this work for solving the optimization problem, finds a solution in reasonable time. This method has the advantage that, compared to gradient descent methods, the number of iterations is constant and does not depend on the input data.

The method of CFC can be seen as applying computed torque control to obtain joint torques and subsequently computing a vector of muscle forces. This latter step utilizes the optimization and the muscle Jacobian, which fully linearizes the system dynamics. Hence the linear system can be stabilized by a PD controller (see Fig. 5.4). The control input f_{ref} satisfies the following equation

$$H\left[P \cdot \Delta q + D \cdot \Delta \dot{q}\right] + C\dot{q} + \tau_G = -L^T \cdot f_{ref} \tag{5.30}$$

which, assuming perfect system models, leads to the linearized closed loop behavior.

$$P \cdot \Delta q + D \cdot \Delta \dot{q} = \ddot{q} \tag{5.31}$$

The transfer function of the closed loop for a single joint $T_f(s)$ can be expressed in the frequency domain for the complex frequency s

$$T_f(s) = \frac{p_i + d_i \cdot s}{p_i + d_i \cdot s + s^2} \tag{5.32}$$

where p_i and d_i denote the i^{th} entry in the diagonal gain matrices $P = diag(p_i)$ and $D = diag(d_i)$, respectively. Stability for this type of system can be proven using methods from linear system theory [28]. Therefore the controller shows global asymptotic stability, assuming (1) perfect system models, (2) quasicontinuous control and (3) perfect low-level force control.

5.2.3 *Computed Actuator Position Control*

By extending the method of computed velocity control to account for the dynamic effects, a third possible controller can be found. Here, the knowledge of the spring stiffness K is used to compute a vector of motor positions to apply a certain force from the current motor position θ and the current force f as follows.

$$\theta_{ref} = \theta + (R_S K)^{-1}(f_{ref} - f) \tag{5.33}$$

The reference force f_{ref} is obtained by the CFC method, hence extending it towards utilizing a low-level actuator position control instead of the force control. This approach has been proposed by Abdallah et al. [1] for a tendon-driven hand and is an extension to the method of computing motor positions for flexible joint robots from a reference torque (see Eq. 4.5). The behavior that is obtained by such a control scheme resembles the Equilibrium Point Hypothesis (EPH) [25] such that the arm is drawn to the equilibrium point that is defined by the reference positions. The equilibrium point is in this case determined by the dynamic model of the robot, the system state and the reference acceleration. Especially when concerned with slow sensor updates or when movements are performed in a FFW manner it is possible that a position based high-level control law shows a better performance than a force based high-level control law.

By utilizing the methods developed in Section 5.2.1, it is possible to compensate for the movement of the joint in a FFW manner, by adding an additional term that adds the change in muscle length in the subsequent control cycle.

$$\theta_{ref} = \theta + (R_S K)^{-1}(f_{ref} - f) + R_S \cdot L\dot{q} \cdot T \tag{5.34}$$

Similar to CFC, the fact that (1) quasicontinuous control, (2) perfect system models and (3) perfect motor position control is assumed, leads to full compensation of the nonlinear dynamics. Therefore, the stability of the proposed controller can be proven for the linearized system, equally to the case of computed force control.

5.3 TRAJECTORY TRACKING

Instead of formulating a regulation controller, often trajectory tracking controllers are proposed [119]. When, instead of a single position, a full trajectory is given, the perfor-

mance of the controller is often considered to be better. Hence, a PD trajectory tracking controller is defined for the linearized system [119],

$$v = \ddot{q}_{ref} + D(\dot{q}_{ref} - \dot{q}) + P \cdot \Delta q \tag{5.35}$$

leading to the following error dynamics

$$\ddot{e} + D \cdot \dot{e} + P \cdot e = 0 \tag{5.36}$$

with $e = \Delta q$, for which stability can be again proven by linear system theory.

Obviously this controller can be implemented for both the computed force, as well as the computed motor position approach, where Eq. 5.35 is applied to either one of the linearized systems to compute either a reference force or a reference actuator position. The fact that the error e converges to zero, proves that also the quaternion difference ΔQ of all spherical joints, converges to $[1 \quad 0 \quad 0 \quad 0]^T$, denoting the zero rotation (see Eq. 5.22).

The trajectory control approach can even be used when the control problem at hand is actually a regulation problem. This can be addressed by transforming the regulation problem into a trajectory control problem, hence computing a trajectory from the current state to the reference state.

$$\ddot{q}_{ref} = P \cdot \Delta q + D \cdot \Delta \dot{q} \tag{5.37}$$

$$\dot{q}_{ref} = \dot{q} + \Delta t \cdot \ddot{q}_{ref} \tag{5.38}$$

$$q_{ref} = q + \Delta t \cdot \dot{q}_{ref} \tag{5.39}$$

To evaluate the trajectory tracking control laws against the regulation controllers, a step function was executed for the Anthrob elbow joint. Here, the two effective muscles, the Brachialis and the Triceps, were utilized with the two available low-level control laws, (1) for the computed force control law and (2) for the computed motor position control law (see Fig. 5.5). Both trajectory tracking controllers perform equally for this experiment and low tracking errors could be achieved. In contrast, the regulation controller with the computed motor position approach performed poorly. This is due to the fact that results depend heavily on the model of the elastic element which was assumed to be a linear spring. However, the Nitrile Butein Rubber (NBR) rings used for the Anthrob muscles exhibit visco-elastic properties. Hence, the computed motor position approach cannot be easily applied for a regulation controller. In any case smoother movements could be achieved by the trajectory tracking controllers.

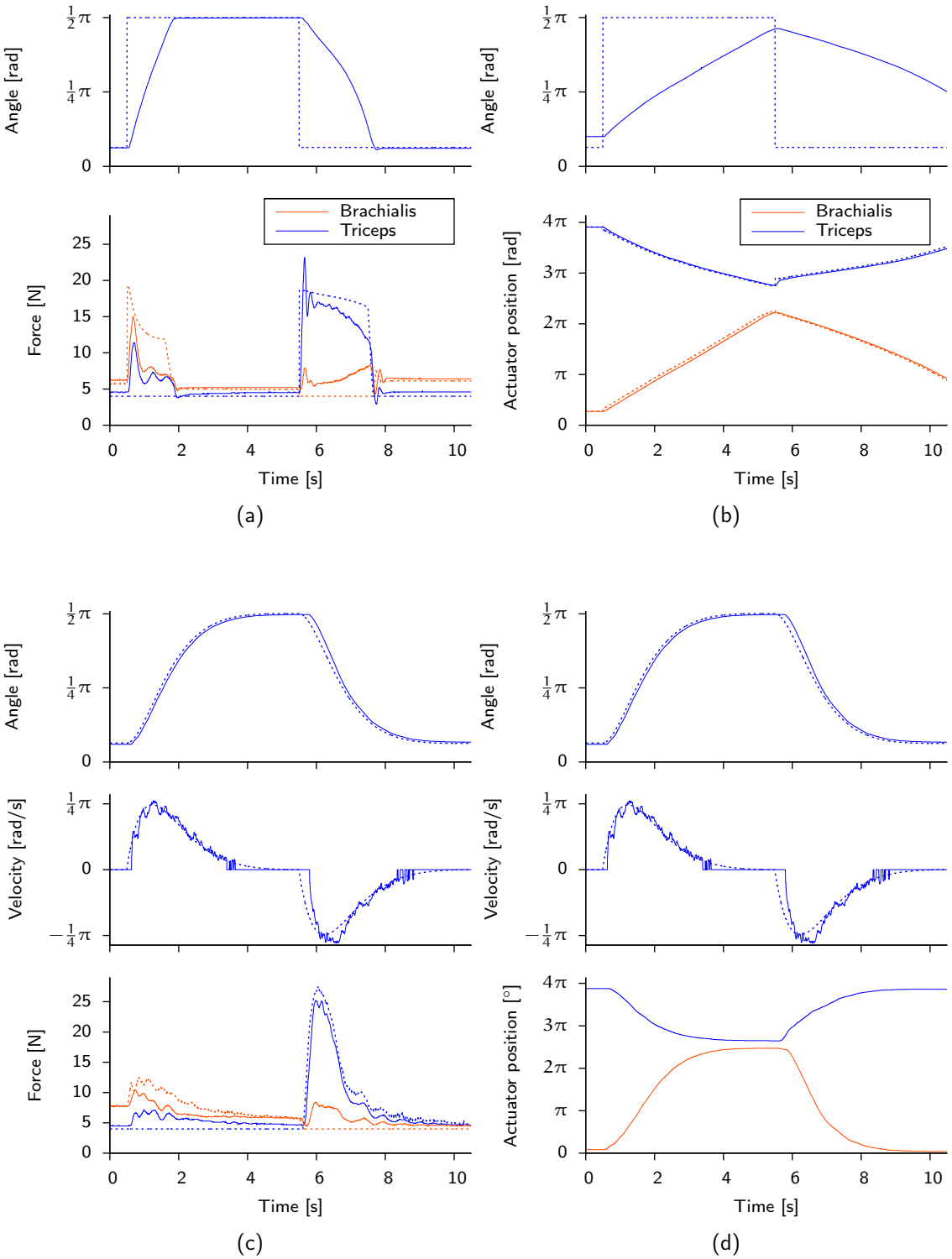


Fig. 5.5: Feedback Linearization Control Comparison. (a) Computed Force Control (regulation), (b) Computed Motor Position Control (regulation) (c) Computed Force Control (trajectory), (d) Computed Motor Position Control (trajectory). A minimum force f_{min} of 4 N is maintained.

6

BACKSTEPPING

Amongst others, the methods described in the previous chapter share a drawback that prevents them from successfully controlling more complex musculoskeletal robots. The proof of stability relies on the assumption of perfect tracking capabilities of the low-level controllers. The reason is that both for the trajectory tracking controllers, as well as for the simpler regulation controllers, the dynamics of the low-level force or actuator position control are neglected. However, this assumption is only valid for control systems, where the dynamics of the low-level controller are one order of magnitude faster than the dynamics of the outer loop. From Fig. 5.2 and Fig. 5.3d it can be seen that this is clearly not the case, here. Rise times in the area of several hundred milliseconds are by far longer than the control period of the dynamics of the high-level controllers. Therefore, the assumptions made for the proof of stability do not hold.

This does not mean that the feedback linearization controllers with underlying tendon force controllers do not deliver reasonable performance in some applications, because clearly they do [1, 50]. First, the high rise times of the force control presented in Section 5.1.2 are caused by the comparably soft materials used for the flexibility in the muscles. The stiffer the muscles are, the faster the rise times, as the actuators do not have to wind up as much tendon. Most of the hands, for which this approach has been developed, were not equipped with additional elastic elements. The flexibility was solely due to the material of the tendon which is stiffer by several orders of magnitude [2]. Therefore, for this type of robot, the feedback linearization controllers perform well. If dealing with more compliant muscles, the problem is particularly apparent, when multi-joint assemblies are considered on which several muscles act in variable directions. For the one joint systems that have been shown in Chapter 5, the movement of the joint will always be in the demanded direction. Therefore, even if muscles pull less than previously calculated by the high-level controller the goal position will eventually be reached. When there are multiple joints, or even a single spherical joint, this is not the case.

In contrast to the control methods developed in the previous chapter, Palli et al. proposed a feedback linearization controller that linearizes the full state of the robot including the muscle kinematics and the actuator dynamics [98, 99]. This type of controller does not rely on a low-level controller and therefore does not exhibit the drawback, mentioned before. However, it was designed for the specific structure of antagonistically actuated revolute joints and cannot be directly applied to the more complicated structure of musculoskeletal robots. Furthermore, the design lacks the possibility of distributing parts of the control, complicating practical implementations in larger setups.

Apart from feedback linearization, there are several techniques that have been used before in robotics to derive non-linear controllers. For flexible-joint robots, the technique of passivity based control has been utilized extensively [5, 17, 97]. The core idea of passivity based control is that a passive system does not gain energy and can therefore be stabilized by any controller that induces damping. For non-passive systems, a controller can be found that converts it into a passive system [60]. However, the method does not give a systematic approach on how to obtain a controller [3].

In contrast, the method of *backstepping* offers a systematic approach to obtain a stable non-linear controller. If the system model can be reformulated to form a system of Ordinary Differential Equations (ODEs) of the following form, this method can be applied [60].

$$\begin{aligned}
\dot{x}_0 &= f_0(x_0) + g_0(x_0)x_1 \\
\dot{x}_1 &= f_1(x_0, x_1) + g_1(x_0, x_1)x_2 \\
\dot{x}_2 &= f_2(x_0, x_1, x_2) + g_2(x_0, x_1, x_2)x_3 \\
&\vdots \\
\dot{x}_{k-1} &= f_{k-1}(x_0, x_1, \ldots, x_{k-1}) + g_{k-1}(x_0, x_1, \ldots, x_{k-1})x_k \\
\dot{x}_k &= f_k(x_0, x_1, \ldots, x_k) + g_k(x_0, x_1, \ldots, x_k)u
\end{aligned} \tag{6.1}$$

Such a system is called a *strict-feedback* system, as the non-linearities f_i and g_i depend only on $x_0, x_1, \ldots, x_i$ whereas g_i has to be invertible for $0 \leqslant i \leqslant k$, over the domain of interest. Furthermore, it is important to note that the i^{th} ODE has to be linear in x_{i+1}.

If this is the case, a recursive procedure can be applied to find a controller for the complete system. This procedure starts by finding a stabilizing controller for the first subsystem, taking x_1 as the virtual control input. Subsequently this controller is stepped back through each of the ODEs, finally producing a controller for the full system. For simplicity, the case of *integrator backstepping* is considered which can be applied to less general systems of the following form,

$$\begin{aligned}
\dot{x}_0 &= f(x_0) + g(x_0)x_1 \\
\dot{x}_1 &= u
\end{aligned} \tag{6.2}$$

where the system state is made up of $x_0 \in \mathbb{R}^n$ and $x_1 \in \mathbb{R}$. If we can find a controller $\phi(x_0)$ that stabilizes the first system equation such that the origin of

$$\dot{x}_0 = f(x_0) + g(x_0)\phi(x_0) \tag{6.3}$$

is asymptotically stable, and there is a Lyapunov function $V_1(x_0)$ which is smooth and positive definite, such that

$$\frac{\partial V_1}{\partial x_0}\left[f(x_0) + g(x_0)\phi(x_0)\right] \leqslant -W(x_0), \quad \forall x_0 \in \mathbb{D} \tag{6.4}$$

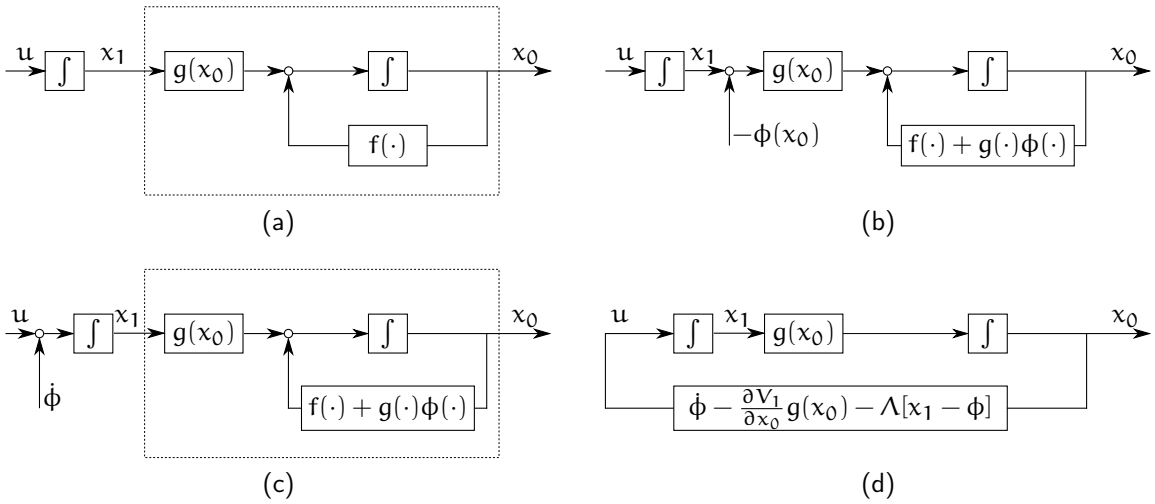


Fig. 6.1: Integrator Backstepping (adapted from [60]). The (a) inner system can be stabilized by (b) a control law ϕ, which is (c) stepped back through the integrator until (d) the final control law is found.

where $W(x_0)$ is positive definite and $\mathbb{D}$ is some domain of interest. Then a controller for the system input u can be found by *backstepping*. By adding and subtracting, Eq. 6.2 can be reformulated.

$$\begin{aligned}\dot{x}_0 &= [f(x_0) + g(x_0)\phi(x_0)] + g(x_0)[x_1 - \phi(x_0)] \\ \dot{x}_1 &= u\end{aligned} \tag{6.5}$$

Based on this, a change of coordinates is defined

$$z_1 = x_1 - \phi(x_0) \tag{6.6}$$

resulting in the following system,

$$\begin{aligned}\dot{x}_0 &= [f(x_0) + g(x_0)\phi(x_0)] + g(x_0)z_1 \\ \dot{z}_1 &= u - \dot{\phi}\end{aligned} \tag{6.7}$$

for which another change of coordinates can be defined.

$$z_2 = u - \dot{\phi} \tag{6.8}$$

Starting from V_1 a Lyapunov function candidate for the full system is found

$$V_2(x_0, x_1) = V_1(x_0) + \frac{1}{2}z_1^2 \tag{6.9}$$

which can be differentiated.

$$\begin{aligned}\dot{V}_2 &= \frac{\partial V_1}{\partial x_0}[f(x_0) + g(x_0)\phi(x_0)] + \frac{\partial V_0}{\partial x_0}g(x_0)z_1 + z_1 z_2 \\ &\leqslant -W(x_0) + \frac{\partial V_1}{\partial x_0}g(x_0)z_1 + z_1 z_2\end{aligned} \tag{6.10}$$

Therefore, a controller for the full system can be found by choosing,

$$z_2 = -\frac{\partial V_1}{\partial x_0} g(x_0) - \Lambda z_1, \quad \Lambda > 0 \tag{6.11}$$

leading to the final Lyapunov function derivative

$$\dot{V}_2 \leqslant -W(x_0) - \Lambda z_1^2 \tag{6.12}$$

which proves asymptotic stability at the origin $(x_0 = 0, z_1 = 0)$. The feedback control law for u can be written as follows, by replacing z_2 in Eq. 6.8.

$$u = \dot{\phi} - \frac{\partial V_1}{\partial x_0} g(x_0) - \Lambda[x_1 - \phi(x_0)] \tag{6.13}$$

The method of *integrator backstepping* can be applied to a wider class of systems, namely the class of *strict-feedback* systems (see Eq. 6.1). In this case, the recursive procedure starts again, with the first subsystem

$$\dot{x}_0 = f_0(x_0) + g_0(x_0) x_1 \tag{6.14}$$

and controller ϕ_1 with respective Lyapunov function V_1 is found for the virtual control input x_1. Subsequently the method of *backstepping* is applied recursively to the i^{th} subsystems with $1 < i \leqslant k+1$. In contrast to the case of *integrator backstepping*, where the second system consists solely of an integrator, here, the controller needs to be stepped back through a more complex system equation (see Eq. 6.1), leading, to the following virtual control law for the i^{th} step.

$$\phi_i = g_{i-1}^{-1} \left[\dot{\phi}_{i-1} - \left(\frac{\partial V_{i-1}}{\partial x_{i-2}} g_{i-2} \right)^T - \Lambda_i (x_{i-1} - \phi_{i-1}) - f_{i-1} \right] \tag{6.15}$$

It can be seen that this controller additionally compensates for the functions g_{i-1} and f_{i-1} which were not existent in the simpler case. A respective Lyapunov function can be stated as follows.

$$V_i = V_{i-1} + \frac{1}{2} z_{i-1}^2 \tag{6.16}$$

Hence, this method describes a systematic way of finding controllers for a class of nonlinear systems. In theory it can be applied up to an arbitrary depth, however, there is the drawback that each controller contains the time derivative of the previous controller (see Eq. 6.15). In [60], this is addressed by differentiating the control law ϕ_{i-1} analytically. However, subsequent differentiations can lead to extremely complex control laws, already after the second or third recursion step. Therefore, methods have to be explored to address this so called explosion of complexity.

The non-linear model developed in Chapter 4 can be rewritten to exhibit the necessary structure to apply *backstepping*.

$$
\begin{aligned}
& H(\alpha)\ddot{q} + C(\alpha, \dot{q})\dot{q} + \tau_G(\alpha) = -L^T(\alpha)x_1 \\
& \dot{x}_1 = \frac{\partial g}{\partial \Delta x}\left[L(\alpha)\dot{q} + R_S x_2\right] && x_1 = f \\
& \dot{x}_2 = \left[-\frac{\mu_v}{r_G c_F J_T} - \frac{c_M^2}{R_A J_T}\right] x_2 - \frac{1}{r_G^2 c_F J_T} R_S x_1 + \frac{c_M}{R_A J_T r_G} v_A && x_2 = \dot{\theta}
\end{aligned}
\tag{6.17}
$$

Here, the first system equation represents the skeletal dynamics and the muscle kinematics and is made up of the pose vector α and the joint velocity $\dot{q}$. It takes the muscle force f as a virtual input which is renamed to x_1. The second ODE represents the muscle dynamics and depends again on the pose vector and the joint velocity, but takes x_2 as a virtual input which is the actuator velocity $\dot{\theta}$. Finally, the last system equation, representing the actuator dynamics, depends on x_1 and x_2 and takes the motor voltage v_A as an input. Therefore, the system fulfills the condition of being a *strict-feedback* system. Furthermore, we have shown that the muscle Jacobian $L(\alpha)$ is invertible by applying the quadratic program (see Eq. 5.28). R_S is a constant and even in the presence of elastic elements with non-linear stress-strain relationship, $\frac{\partial g}{\partial \Delta x}$ will not evaluate to zero, similar to the constant $\frac{c_M}{R_A J_T r_G}$. Hence, the system dynamics fulfill the requirements for *backstepping* to be applicable.

For flexible joint robots, the control law is either stated based on the joint torque or the actuator position. An example for the first one has been shown by Albu-Schäffer and Hirzinger [4] and for the second by Slotine and Li [121]. It is possible to rewrite the system model in Eq. 6.17 to use the actuator position as a state.

$$
H(\alpha)\ddot{q} + C(\alpha, \dot{q})\dot{q} + \tau_G(\alpha) = -L^T(\alpha) \cdot g(R_S \cdot \theta + l(\alpha) - l_0) \tag{6.18}
$$

In this formulation, it is quite obvious that the first subsystem is not necessarily linear in the actuator position θ, due to the non-linear spring. Therefore, the method of *backstepping* is only applicable to such a system if the assumption of linear stress-strain relationship is made,

$$
H(\alpha)\ddot{q} + C(\alpha, \dot{q})\dot{q} + \tau_G(\alpha) - L^T(\alpha)KR_S\left[l(\alpha) - l_0\right] = -L^T(\alpha)KR_S \cdot \theta \tag{6.19}
$$

where K denotes the spring stiffness. The more general solution is therefore, to use the approach of employing the muscle force as a system state.

Oh and Lee [95] designed a non-linear controller for trajectory tracking of flexible joint robots, based on the method of *backstepping*. Several extensions were necessary for the application to the field of musculoskeletal robots which are introduced in the following section. Subsequently, the extension of Dynamic Surface Control (DSC) is applied to solve the problem of explosion of complexity which arises due to the appearance of consecutive differentiation of the controllers.

6.1 BASIC BACKSTEPPING

In the following three sections, controllers for the cascading subsystems are developed, iteratively. Starting with the skeletal dynamics, where an initial controller and corresponding Lyapunov function is chosen. Based on this, controllers for the subsystems of the muscle dynamics, and finally the actuator dynamics are obtained.

6.1.1 *Skeletal Dynamics*

The first step is to choose a controller for the first subsystem in Eq. 6.17 which takes the muscle force as an input. For flexible joint robots a trajectory tracking controller has been proposed by Albu-Schäffer and Hirzinger [4],

$$\begin{aligned}\tau_{\text{ref}} =& H\left[\ddot{q}_d - \Lambda_1(\dot{q} - \dot{q}_d)\right] + C\left[\dot{q}_d - \Lambda_1(q - q_d)\right] \\ &+ \tau_G - K_d\left[\dot{q} - \dot{q}_d + \Lambda_1(q - q_d)\right]\end{aligned} \tag{6.20}$$

where K_d and Λ_1 are positive definite matrices, denoting control gains. Utilizing the muscle Jacobian and an optimization based on the techniques developed in Section 5.2.2, this controller can be directly applied for musculoskeletal robots,

$$\begin{aligned}\phi_1 = -L^{T+}\{&H\left[\ddot{q}_d + \Lambda_1(\dot{q}_d - \dot{q})\right] + C\left[\dot{q}_d + \Lambda_1 \Delta q\right] \\ &+\tau_G - K_d \cdot r\}\end{aligned} \tag{6.21}$$

while L^{T+} denotes a pseudo inverse of the muscle Jacobian and Δq the angle error based on Eq. 5.23 which describes the angle offset in the presence of single DoF joints as well as spherical joints. Note that Δq defines the difference between the reference angle q_d and q. Hence the sign before Δq switched in Eq. 6.21. To ensure all positive muscle forces, first the reference torque τ_{ref} is computed, and subsequently the quadratic program, defined in Eq. 5.28 is executed to obtain the reference force vector ϕ_1. The tracking error r is defined as follows.

$$r = \dot{q} - \dot{q}_d - \Lambda_1 \Delta q$$

Hence, by proving convergence of r to zero, both the error for the reference joint velocity, as well as the error in the joint position Δq is proven to converge to zero. By utilizing the definition of the quaternion error function Eq. 5.23 it has already been shown that all joint angles, both for revolute, as well as spherical joints, converge to the zero rotation (see Eq. 5.22).

The definition of the first change of coordinates shows z_1 as the difference between the reference force ϕ_1 and the force. Therefore it is a measure of the error of the underlying force control.

$$z_1 = x_1 - \phi_1 \tag{6.22}$$

Stability of the given control law can be proven, assuming perfect system models. Hence, the closed loop dynamics evaluate to

$$H\dot{r} + Cr + K_d r = -L^T z_1 \tag{6.23}$$

for which a Lyapunov function candidate is chosen.

$$V_1 = \frac{1}{2} r^T H r \tag{6.24}$$

Due to the property of the inertia Matrix being positive definite (see Section 4.1.2), it can be seen that $V(r) > 0$ for any $r \neq 0$. Its derivative satisfies the following equation.

$$\begin{aligned}\dot{V}_1 =& \frac{1}{2} r^T \dot{H} r + r^T H \dot{r} \\ =& \frac{1}{2} r^T \dot{H} r + r^T \left[-Cr - K_d r - L^T z_1 \right] \\ =& \frac{1}{2} r^T \left[\dot{H} - 2C \right] r - r^T K_d r - r^T L^T z_1 \end{aligned} \tag{6.25}$$

It can be shown that $[\dot{H} - 2C]$ is skew symmetric (see Eq. 4.13) and the derivative finally evaluates to the following relationship.

$$\dot{V}_1 = -r^T K_d r - r^T L^T z_1 \tag{6.26}$$

It can be seen that the Lyapunov function derivative contains a term that depends on the error of the force control z_1 and the tracking error r. This term is therefore a function of the error of the low-level muscle force controller. In the subsequent iterative steps, suitable compensation for this term is introduced. It is obvious that, provided a low-level controller drives the error z_1 to zero, the derivative of the Lyapunov function is negative definite for positive definite control gains K_d. Hence, the state of r is equally driven to zero by the chosen controller.

6.1.2 *Muscle Dynamics*

The muscle dynamics consist of a model of the change in muscle length, due to the movement of the joints, and a model of the elastic element. Here, the partial derivative $\frac{\partial g}{\partial \Delta x}$ can be seen as the current spring stiffness K which can be either written as a function of the spring deflection Δx or as function of the force.

$$K(x_1) = \frac{\partial g}{\partial \Delta x} \tag{6.27}$$

Therefore, the second system equation from Eq. 6.17 can be rewritten, leading to the form presented in Eq. 6.1,

$$\begin{aligned}\dot{x}_1 &= K(x_1) \cdot L(\alpha)\dot{q} + K(x_1) \cdot R_S x_2 \\ &= f_1(\alpha, \dot{q}, x_1) + g_1(x_1) x_2 \end{aligned} \tag{6.28}$$

which can be introduced into the differentiated definition of z_1 (see Eq. 6.22).

$$\dot{z}_1 = \dot{x}_1 - \dot{\phi}_1 = f_1 + g_1 x_2 - \dot{\phi}_1 \tag{6.29}$$

Similar to the first change of coordinates, the definition of z_2 is an error function for the next low-level controller which controls the actuator velocity.

$$z_2 = x_2 - \phi_2 \tag{6.30}$$

According to the method of *backstepping*, the controller ϕ_i contains a term that depends on the previous Lyapunov function V_{i-1} which is applied to the Lyapunov function for the skeletal dynamics in Eq. 6.24 as follows.

$$\frac{\partial V_1}{\partial x_0} g_0 = -r^T H \cdot H^{-1} L^T = -(Lr)^T \tag{6.31}$$

Hence a control law for the muscle dynamics is obtained (see also Eq. 6.15).

$$\phi_2 = g_1^{-1} \left[\dot{\phi}_1 + Lr - \Lambda_2 z_1 - f_1 \right] \tag{6.32}$$

The closed loop dynamics can be evaluated by introducing this control law into Eq. 6.29, by means of the second change of coordinate from Eq. 6.30.

$$\begin{aligned} \dot{z}_1 &= f_1 + g_1 [z_2 + \phi_2] - \dot{\phi}_1 \\ &= g_1 z_2 - \Lambda_2 z_1 + Lr \end{aligned} \tag{6.33}$$

Similarly, the recursive procedure, provides a Lyapunov function candidate.

$$V_2 = V_1 + \frac{1}{2} z_1^T z_1 \tag{6.34}$$

Finally, the closed loop dynamics from Eq. 6.33 are introduced into the Lyapunov function derivative.

$$\begin{aligned} \dot{V}_2 &= \dot{V}_1 + z_1^T \dot{z}_1 \\ &= -r^T K_d r - z_1^T \Lambda_2 z_1 + z_1^T g_1 z_2 \end{aligned} \tag{6.35}$$

Similar to $\dot{V}_1$, the Lyapunov function derivative in Eq. 6.35 contains a mixed term with the error function for the next controller level z_2. However, the term depending on z_1 was eliminated by the compensation term in the muscle dynamics controller, as expected. The Lyapunov function derivative is negative definite, provided the error term z_2 is driven to zero, proving convergence to zero both for the state of r, and the state of z_1.

6.1.3 *Actuator Dynamics*

Rewriting the third system equation from Eq. 6.17, leads to the equation for the actuator dynamics. With the voltage v_A, as an input, a controller for the actuator velocity x_2 is to be found for the following system.

$$\begin{aligned}\dot{x}_2 &= \left[-\frac{\mu_v}{r_G c_F J_T} - \frac{c_M^2}{R_A J_T}\right] x_2 - \frac{1}{r_G^2 c_F J_T} R_S x_1 + \frac{c_M}{R_A J_T r_G} v_A \\ &= f_2(x_1, x_2) + g_2 v_A \end{aligned} \tag{6.36}$$

Eq. 6.36 denotes the scalar case for a single actuator. Due to the fact that at the level of the actuator dynamics there is no cross-dependence between the muscles, this controller can be developed for the scalar case and instantiated for each of the muscles individually. Similar to the previous step, this equation of motion can be introduced into the differentiated definition of z_2 (see Eq. 6.30)

$$\dot{z}_2 = \dot{x}_2 - \dot{\phi}_2 = f_2 + g_2 v_A - \dot{\phi}_2 \tag{6.37}$$

for which a controller can be found.

$$\phi_3 = g_2^{-1}\left[\dot{\phi}_2 - g_1 z_1 - \Lambda_3 z_2 - f_2\right] \tag{6.38}$$

The closed loop dynamics are obtained by replacing the input voltage in Eq. 6.36 by this control law.

$$\begin{aligned}\dot{z}_2 &= f_2 + \dot{\phi}_2 - f_2 - \Lambda_3 z_2 - g_1 z_1 - \dot{\phi}_2 \\ &= -\Lambda_3 z_2 - g_1 z_1 \end{aligned} \tag{6.39}$$

Again, a Lyapunov function candidate is provided

$$V_3 = V_2 + \frac{1}{2} z_2^T z_2 \tag{6.40}$$

which can be differentiated and merged with the closed loop dynamics from Eq. 6.39.

$$\begin{aligned}\dot{V}_3 &= \dot{V}_2 + z_2^T \dot{z}_2 \\ &= -r^T K_d r - z_1^T \Lambda_2 z_1 + z_1^T g_2 z_2 - z_2^T \Lambda_3 z_2 - z_2 g_2^T z_1 \\ &= -r^T K_d r - z_1^T \Lambda_2 z_1 - z_2 \Lambda_3 z_2 \end{aligned} \tag{6.41}$$

Finally a Lyapunov function derivative without mixed terms is obtained. The controller for the last step of the procedure, compensates for the term $z_1 g_1 z_2$ which was still present in $\dot{V}_2$, hence eliminating it in $\dot{V}_3$.

6.1.4 *Controller Discussion*

The previous three sections have produced three control laws for the three subsystems. While the first one is similar to trajectory tracking controllers for flexible joint roots, with the extension of the muscle Jacobian, the latter two define a novel force controller with a compensation for muscle length changes, due to the changes in the joint angles. Furthermore, the muscle force controller is able to handle non-linear elastic elements.

$$\phi_1 = -L^{T+}(\alpha)\{H(\alpha)\left[\ddot{q}_d + \Lambda_1(\dot{q}_d - \dot{q})\right] + C(\alpha, \dot{q})\left[\dot{q}_d + \Lambda_1 \Delta q\right] + \tau_G(\alpha) - K_d\left[\dot{q} - \dot{q}_d - \Lambda_1 \Delta q\right]\}$$

$$\phi_2 = g_1^{-1}(f)\left\{\dot{\phi}_1 + L(\alpha)\left[\dot{q} - \dot{q}_d - \Lambda_1 \Delta q\right] - \Lambda_2\left[f - \phi_1\right] - f_1(\alpha, \dot{q}, f)\right\}$$

$$\phi_3 = g_2^{-1}\left\{\dot{\phi}_2 - g_1(f)\left[f - \phi_1\right] - \Lambda_3\left[\dot{\theta} - \phi_2\right] - f_2(f, \dot{\theta})\right\}$$

Stability of the proposed controller can be easily proven through Lyapunov's stability theorem, where the Lyapunov function candidate is given as follows.

$$V = V_3 = \frac{1}{2} r^T H r + \frac{1}{2} z_1^T z_1 + \frac{1}{2} z_2^T z_2 \tag{6.42}$$

By differentiation, the Lyapunov function derivative is obtained

$$\dot{V} = \dot{V}_{bs} = \dot{V}_3 = -r^T K_d r - z_1^T \Lambda_2 z_1 - z_2 \Lambda_3 z_2 \tag{6.43}$$

which is negative definite, provided that K_d, Λ_1 and Λ_2 are chosen to be positive definite matrices. These three controller gain matrices are usually diagonal matrices with positive elements, where $K_d \in \mathbb{R}^{n \times n}$ and $\Lambda_1, \Lambda_2 \in \mathbb{R}^{m \times m}$. Therefore, it can be concluded that the developed controller is asymptotically stable under the assumption of quasicontinuous control and perfect system models.

The controller can be distributed such that the muscle force controller, consisting of ϕ_2 and ϕ_3, is implemented on the distributed Electronic Control Units (ECUs) (see Section 3.5) and ϕ_1 is implemented on the central controller. This is possible, since only the high-level controller needs to take cross-dependencies between the muscles into account. On the lower levels, the controllers are independent of each other. Therefore, only the result of ϕ_1 and a FFW part is needed for computing ϕ_2. This FFW part is defined as follows and needs to be evaluated centrally for each of the muscles and communicated to the low-level controllers via the bus system.

$$FFW = L(\alpha) r - f_1(\alpha, \dot{q}, f) \tag{6.44}$$

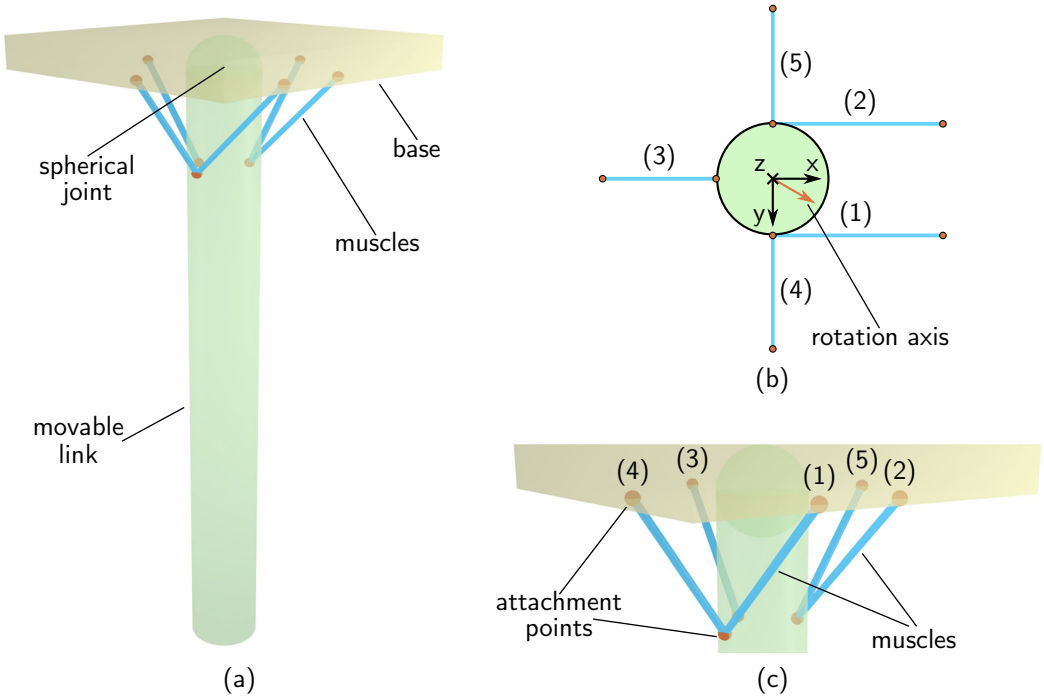


Fig. 6.2: Simulation Model. It consists of a base and a movable link that is connected via a spherical joint. The system is actuated by five muscles with equal parameters. A (a) rendering of the full model is shown, as well as (b) a top view, and (c) a close-up of the muscle attachment points.

This provides a similar structure, as for the feedback linearization controllers, where low-level controllers can run at a much higher frequency than the central controller.

An evaluation of this control scheme was performed, utilizing the simulation of a spherical joint that is spanned by five muscles (see Fig. 6.2c). The muscles were attached asymmetrically such that muscles (1) and (2) are positioned excentrically, to allow for rotational movements (see Fig. 6.2b). The simulation model was implemented in MATLAB/Simulink, utilizing the full dynamic model of muscles with linear springs (see Section 4.2) and of the movable link. The muscle kinematics were simulated by means of computing muscle lever arms for given joint angles, assuming straight line muscles (see Section 4.3). For results to be as realistic as possible, muscle attachment points were chosen to resemble the order of magnitude for the lever arms as in the Anthrob (see Chapter 3). A friction model for the joint friction, as well as friction due to the tendon routing was introduced. While the first was implemented by the extended Stribeck model, the latter was kept more simple by assuming Coulomb friction which depends only on the transmitted force (see Section A.5). Friction parameters were initially chosen such that friction losses were negligible, but were increased for later experiments to evaluate the

robustness of the developed controllers. For a full display of the model parameters, refer to Table B.3.

For the implementation in simulation, there are two possibilities for handling the controller derivatives $\dot{\phi}_{1/2}$. Either, the controllers are differentiated analytically or numerical differentiation is performed at runtime. While the first leads to the explosion of complexity problem (see Section 6.1), the latter led to numerical instabilities. Hence, the two controller derivatives were set to zero, to numerically stabilize the system. This *backstepping* controller was implemented with the control frequencies that are possible to achieve with the control architecture proposed in Section 3.5, resulting in 200 Hz for the execution of ϕ_1 and 1 kHz for the low-level control, comprising ϕ_2 and ϕ_3. This control law evaluation was performed by applying a reference trajectory, leading to a movement of the spherical joint of 45° around a fixed axis in the X-Y plane. This axis was chosen to apply asymmetric loads to the muscles, by rotating the X-Axis around Z, by 30° (see Fig. 6.2b). The resulting movement was performed twice in a row, while the resulting trajectory and the tracking errors were recorded for all three levels of the control law.

The performance of the distributed low-level control laws can be observed in Fig. 6.3. The muscles (1), (2) and (5) were responsible for lifting the link, hence resulting in positive motor velocities during the step up and negative motor velocities during the step down. Slight deviation between force reference and the system output e. g. for muscle (5) is apparent throughout the trajectory. The Root Mean Square Error (RMSE) for this muscle can be given as 0.29 N. It becomes clear that omitting the controller derivatives from the implementation led to chattering of the force reference, as well as the motor velocity reference. The position error for the spherical joint is given by Δq (see Eq. 5.23). A mean angle error over the full trajectory can be given as $0.0047\,\mathrm{rad}$, where the angle error is the Euclidean norm $||\Delta q||$, such that it denotes the absolute deviation of the angle in any direction.

From the simulated results it can be seen that the *backstepping* method leads to a stabilizing controller that can be used to control a robot with spherical joints. However, the necessary simplifications led to relatively poor low-level control results even for perfect system models. In the following section, the problem of the explosion of complexity is addressed.

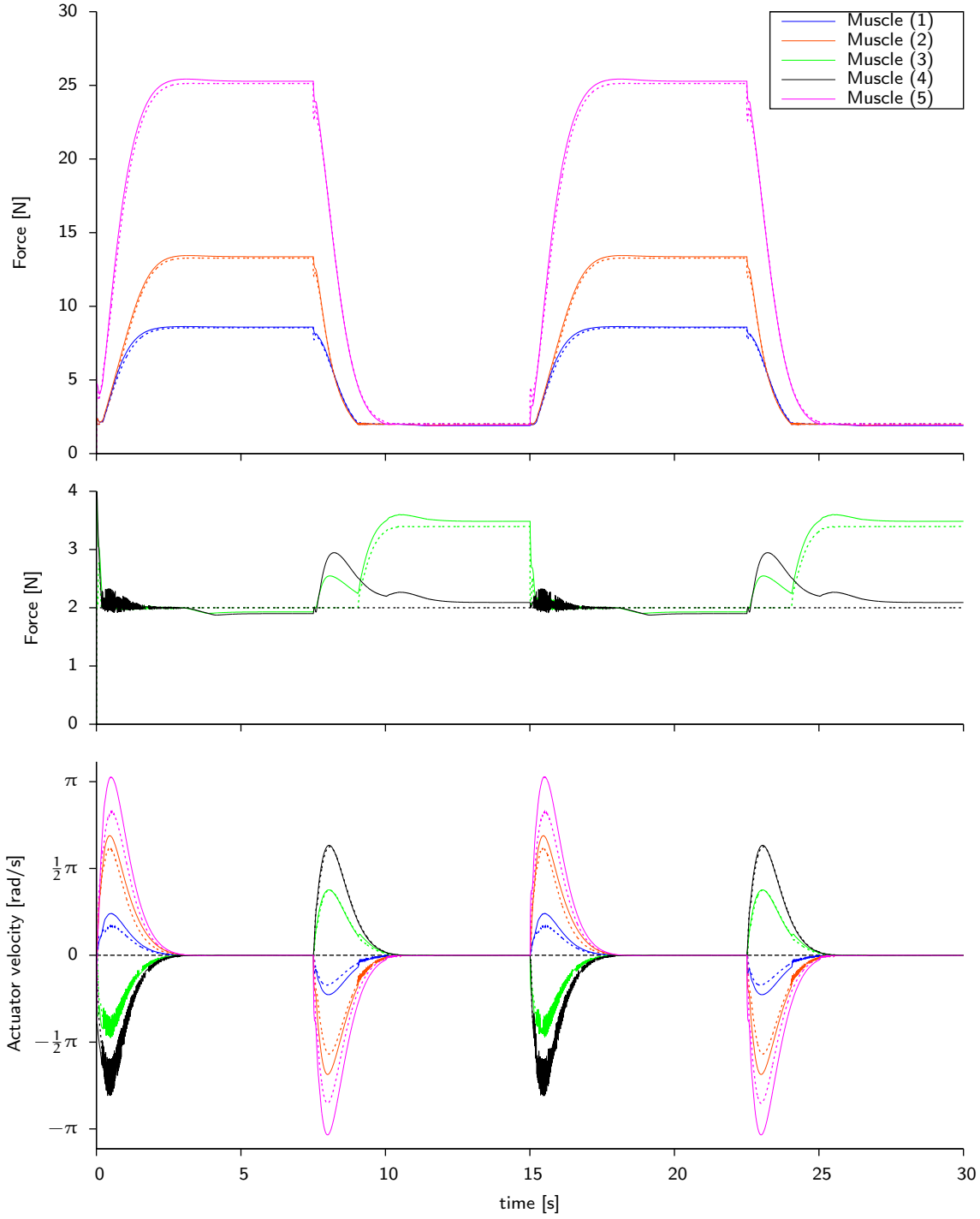


Fig. 6.3: Backstepping Low-Level Control. The reference (dashed) is shown together with the system output, both for the active forces (top), the inactive forces (center), and the actuator velocity control law (bottom). Forces were controlled to maintain a minimum of 2 N. The trajectory is shown twice for a step up and down.

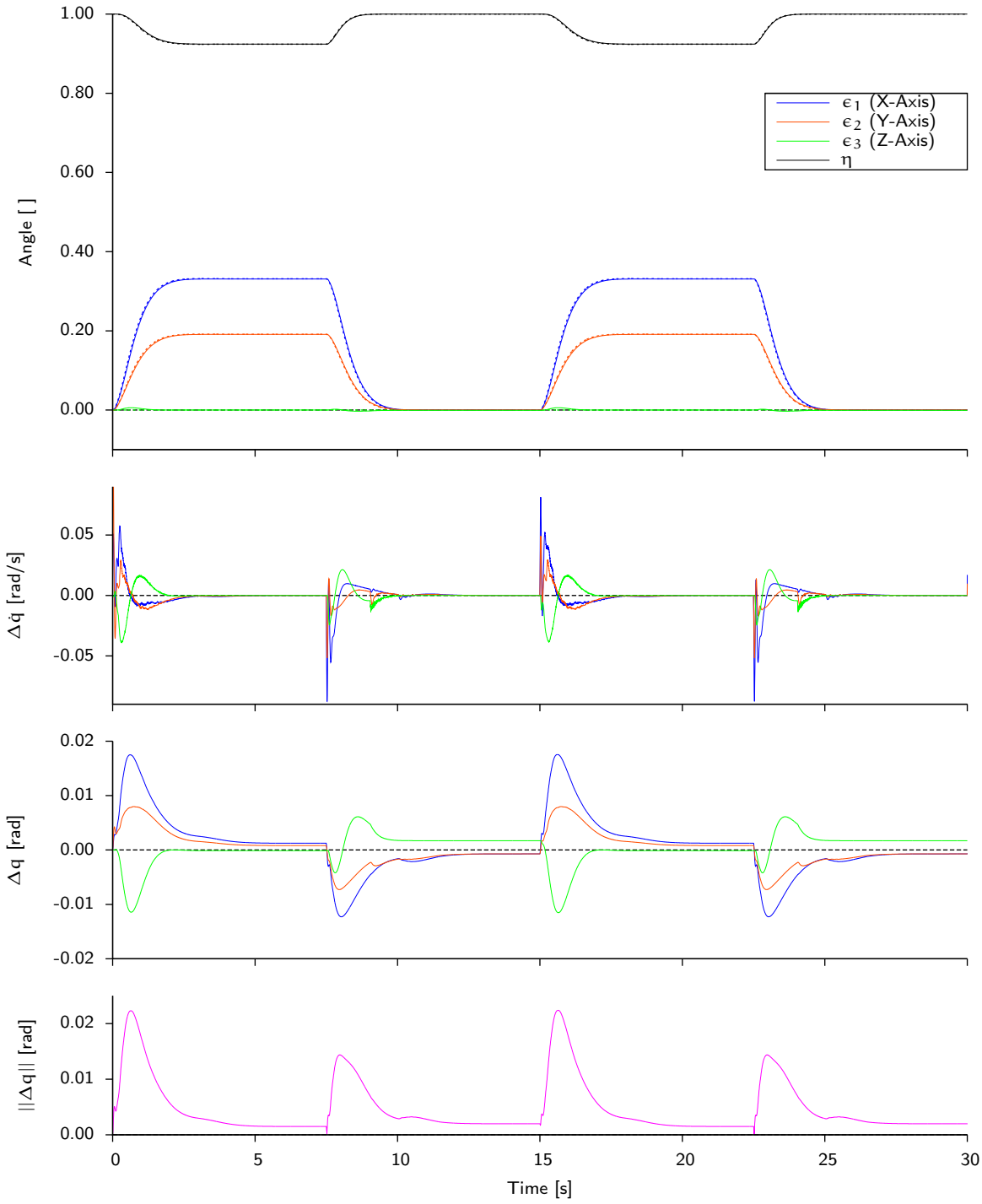


Fig. 6.4: Backstepping High-Level Control. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in velocity $\Delta\dot{q}$, the error in position Δq, and the absolute angle error (bottom).

6.2 DYNAMIC SURFACE CONTROL

To improve the performance of the previous controller without having to deal with the problem of the explosion of complexity, the technique of Dynamic Surface Control (DSC) can be utilized. It is an extension to *backstepping* that adds a set of first order low-pass filters in between each of the steps of the individual controllers. This technique has been developed by Swaroop et al. [125] to solve the previously mentioned problem, enabling practical controller implementation which was complicated by the first order differentiation in backstepping. Furthermore, it has been shown that controllers developed with the technique of DSC feature guaranteed tracking errors that are confined to a ball of adjustable radius, i. e. it shows Uniformly Ultimately Bounded (UUB) stability. In the following, this technique is applied to the problem of musculoskeletal control, based on the controller developed in the previous section.

Each of the first order filters is introduced to filter the signal of the controller output. Therefore, each of the filters is defined such that

$$\mu_i \dot{s}_i + s_i = \phi_i \qquad i = \{1, \dots, k\} \tag{6.45}$$

whereas μ_i is the filter constant, s_i is an additional coordinate, denoting the output of the filter, and k is the depth of the *strict-feedback* system (see Eq. 6.1). Therefore, the previously defined coordinates z_i are redefined,

$$z_i = x_i - s_i \tag{6.46}$$

utilizing the filtered s_i, instead of ϕ_i. The now filtered derivative of the controller $\dot{s}_i$, can be replaced by the following term,

$$\dot{s}_i = \frac{\phi_i - s_i}{\mu_i} \tag{6.47}$$

stabilizing the numerical performance of the online derivative. Note, that low-pass filtering the controller signals decreases the reaction times, while improving numerical stability. Therefore, choosing the time constants μ_i poses a trade-off between speed and stability.

6.2.1 *Application to the Control Problem*

Two first order filters are applied, one between each of the controllers previously derived (see Fig. 6.5). While the control law for the skeletal dynamics in Eq. 6.21 is left unchanged, the first filter is introduced by substituting the filter dynamics (see Eq. 6.45) into the definition of s_1 in Eq. 6.46

$$x_1 = z_1 + s_1 = z_1 + \phi_1 - \mu_1 \dot{s}_1 \tag{6.48}$$

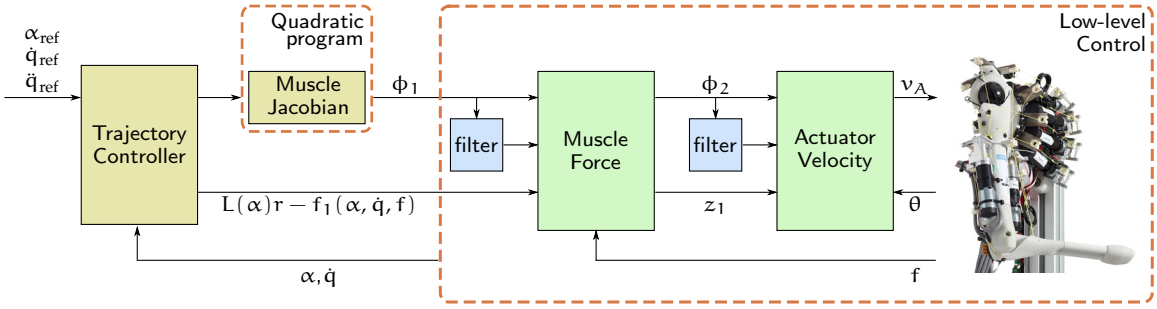


Fig. 6.5: Block Diagram of Dynamic Surface Control. The controller consists of three levels, starting with the controller for the skeletal dynamics, comprising the trajectory controller and the muscle Jacobian. The other two levels are the muscle force and the actuator velocity controllers. The filters are added to obtain filtered values of controllers wherever the controller derivatives $\dot{\phi}_i$ are needed.

which affects the closed loop dynamics. An additional term for the state of the filter is added.

$$H\dot{r} + Cr + K_d r = -L^T [z_1 - \mu_1 \dot{s}_1] \tag{6.49}$$

The controller for the muscle dynamics from Eq. 6.32 is found by replacing the problematic derivative $\dot{\phi}_1$ with the filtered $\dot{s}_1$,

$$\phi_2 = g_1^{-1} [\dot{s}_1 + Lr - \Lambda_2 z_1 - f_1] \tag{6.50}$$

while the closed loop dynamics for the muscle can be obtained by adding the filter to the muscle dynamics in Eq. 6.28.

$$\begin{aligned} \dot{z}_1 &= f_1 + g_1 [z_2 + \phi_2 - \mu_2 \dot{s}_2] - \dot{s}_1 \\ &= g_1 z_2 - g_1 \mu_2 \dot{s}_2 - \Lambda_2 z_1 + Lr \end{aligned} \tag{6.51}$$

Finally, the controller for the actuator dynamics can be stated by the same method,

$$\phi_3 = g_2^{-1} [\dot{s}_2 - g_1 z_1 - \Lambda_3 z_2 - f_2] \tag{6.52}$$

leading to the following closed loop system, by substitution into the actuator dynamics in Eq. 6.36.

$$\dot{z}_2 = -g_1 z_1 - \Lambda_3 z_2 \tag{6.53}$$

The application of DSC changes little in the matter of the controller definition, apart from replacing each controller ϕ_i with the filter output s_i. However, the dynamics are changed on each level, by adding the extra state of the filter which affects the proof of stability.

6.2.2 *Stability Proof*

Due to the inclusion of the filter terms, the closed loop system dynamics, as well as the proof of stability are of increased complexity. The Lyapunov function candidate needs to be extended by an additional term per filter which describes the error due to the filter [87, 125]. It can be defined as

$$e_i = s_i - \phi_i \tag{6.54}$$

which can be directly substituted into the derivative of s_i (see Eq. 6.47) as follows.

$$\dot{s}_i = \frac{\phi_i - s_i}{\mu_i} = -\frac{e_i}{\mu_i} \tag{6.55}$$

By differentiation, the derivative of the filter error $\dot{e}_i$ is obtained,

$$\dot{e}_i = \dot{s}_i - \dot{\phi}_i = -\frac{e_i}{\mu_i} + \xi_i(\dots) \tag{6.56}$$

where ξ_i is a bounded continuous function [125, 136, 147] of the system states, their derivatives, the controllers, and the reference trajectory. As the controller is only operated within certain bounds, there is an upper bound to $\dot{e}_i$. A Lyapunov function candidate is chosen [87],

$$V = \frac{1}{2} r^T H r + \frac{1}{2} z_1^T z_1 + \frac{1}{2} z_2^T z_2 + \frac{k_e}{2} e_1^T e_1 + \frac{k_e}{2} e_2^T e_2 \tag{6.57}$$

where k_e is a design parameter. Recalling the Lyapunov function from Section 6.1, the derivative of V can be obtained by differentiation and introduction of the closed loop dynamics from Eq. 6.49, Eq. 6.51 and Eq. 6.53. Apart from the filter error terms, two additional terms depending on the filter derivatives $\dot{s}_i$ come into play and can be replaced by Eq. 6.55. Furthermore, Eq. 6.56 is introduced for $\dot{e}_1$ and $\dot{e}_2$.

$$\begin{aligned}
\dot{V} =& - r^T K_d r - z_1^T \Lambda_2 z_1 - z_2^T \Lambda_3 z_2 \\
&+ r^T L^T \mu_1 \dot{s}_1 - z_1^T g_1 \mu_2 \dot{s}_2 \\
&+ k_e e_1^T \dot{e}_1 + k_e e_2^T \dot{e}_2 \\
=& - r^T K_d r - z_1^T \Lambda_2 z_1 - z_2^T \Lambda_3 z_2 \\
&- r^T L^T e_1 + z_1^T g_1 e_2 \\
&- \frac{k_e}{\mu_1} e_1^T e_1 - \frac{k_e}{\mu_2} e_2^T e_2 + k_e e_1^T \xi_1(\dots) + k_e e_2^T \xi_2(\dots) \\
=& + \dot{V}_{bs} + \dot{V}_{dsc}
\end{aligned} \tag{6.58}$$

For simplicity we define $\dot{V}_{dsc}$ as follows,

$$\begin{aligned}
\dot{V}_{dsc} =& - r^T L^T e_1 + z_1^T g_1 e_2 \\
&- \frac{k_e}{\mu_1} e_1^T e_1 - \frac{k_e}{\mu_2} e_2^T e_2 + k_e e_1^T \xi_1(\dots) + k_e e_2^T \xi_2(\dots)
\end{aligned}$$

which is the part of $\dot{V}$ which differs from the Lyapunov function derivative $\dot{V}_{bs}$ in Eq. 6.43. At this point Young's inequality [10, 148] is used to state an upper bound for $\dot{V}_{dsc}$. It is defined as,

$$ab \leqslant \frac{a^2}{2c} + \frac{cb^2}{2} \tag{6.59}$$

whereas the positive constant c can be redefined as $2c_j$ (with $j = \{1, \dots, 4\}$) to be applied four times to $\dot{V}_{dsc}$, leading to the following inequality.

$$\begin{aligned}\dot{V}_{dsc} \leqslant &+ \frac{r^T r}{4c_1} + c_1 e_1^T L L^T e_1 + \frac{z_1^T z_1}{4c_2} + c_2 e_2^T g_1^T g_1 e_2 \\ &- \frac{k_e}{\mu_1} e_1^T e_1 - \frac{k_e}{\mu_2} e_2^T e_2 \\ &+ \frac{e_1^T e_1}{4c_3} + c_3 k_e^2 \xi_1^T(\dots)\xi_1(\dots) + \frac{e_2^T e_2}{4c_4} + c_4 k_e^2 \xi_2^T(\dots)\xi_2(\dots)\end{aligned} \tag{6.60}$$

From the definition of $\xi_i(\dots)$ (see Eq. 6.56) it can be seen that $|\xi_i(\dots)|$ has an upper bound that is defined as M_i [87]. For simplicity, $c_{3/4}$ are chosen to be $\frac{1}{4}$. Therefore $\dot{V}$ satisfies the following inequality,

$$\begin{aligned}\dot{V} \leqslant &- r^T \left[K_d - I_n \frac{1}{4c_1} \right] r - z_1^T \left[\Lambda_2 - I_m \frac{1}{4c_2} \right] z_1 - z_2^T \Lambda_3 z_2 \\ &- e_1^T \left[I_m \frac{k_e}{\mu_1} - c_1 L L^T - I_m \right] e_1 - e_2^T \left[I_m \frac{k_e}{\mu_2} - c_2 g_1^T g_1 - I_m \right] e_2 \\ &+ \frac{1}{4} k_e^2 M_1^T M_1 + \frac{1}{4} k_e^2 M_2^T M_2\end{aligned} \tag{6.61}$$

where $c_{1/2}$ and k_e can be chosen arbitrarily in $\mathbb{R}_{>0}$. According to Lyapunov's stability theorem there exist gains such that UUB stability of the closed loop system can be guaranteed [87]. This is true if the gains meet the following inequalities.

$$I_m \mu_1 \leqslant k_e \cdot (c_1 L L^T + I_m)^{-1} \tag{6.62}$$

$$I_m \mu_2 \leqslant k_e \cdot (c_2 g_1^T g_1 + I_m)^{-1} \tag{6.63}$$

$$K_d \geqslant I_n \frac{1}{4c_1} \tag{6.64}$$

$$\Lambda_2 \geqslant I_m \frac{1}{4c_2} \tag{6.65}$$

$$\Lambda_3 \geqslant 0 \tag{6.66}$$

For very small μ_i the design parameter k_e can be chosen to be as small as necessary, without violating Lyapunov's stability theorem, hence making the last two terms of Eq. 6.61 arbitrarily small. However, this impairs at the same time the numeric stability, as $\dot{s}_i$ becomes sensitive to changes (see Eq. 6.47). Ultimately, as $\mu_i \to 0$, this obviously leads to the case of basic *backstepping*. In practice, a trade-off between numeric stability and a low tracking error has to be found.

6.2.3 *Controller Discussion*

The control laws, developed by the method of DSC are structured similarly to the basic *backstepping* controller. However, numeric stability due to the differentiation can be greatly improved by finding suitable filter parameters μ_i. The separate implementation of a high-level controller and distributed muscle force controllers can be continued, while filters are implemented as a part of the low-level control running at higher frequencies (see Fig. 6.5).

$$\phi_1 = -L^{T+}(\alpha)\{H(\alpha)\left[\ddot{q}_d + \Lambda_1(\dot{q}_d - \dot{q})\right] + C(\alpha,\dot{q})\left[\dot{q}_d + \Lambda_1 \Delta q\right] + \tau_G(\alpha) - K_d\left[\dot{q} - \dot{q}_d - \Lambda_1 \Delta q\right]\}$$

$$\phi_2 = g_1^{-1}(f)\left\{\frac{\phi_1 - s_1}{\mu_1} + L(\alpha)\left[\dot{q} - \dot{q}_d - \Lambda_1 \Delta q\right] - \Lambda_2\left[f - s_1\right] - f_1(q,\dot{q},f)\right\}$$

$$\phi_3 = g_2^{-1}\left\{\frac{\phi_2 - s_2}{\mu_2} - g_1(f)\left[f - s_1\right] - \Lambda_3\left[\dot{\theta} - s_2\right] - f_2(f,\dot{\theta})\right\}$$

An evaluation via simulation of the proposed controller was performed with the same setup as for the basic *backstepping* controller, featuring a spherical joint and five muscles (see Section 6.1.4). Due to the utilization of the filtered controller derivatives $\dot{\phi}_{1/2}$, the oscillations in the low-level controllers could be removed for all five muscles, compared to the basic *backstepping* (see Fig. 6.6). The high-level trajectory tracking performance has improved slightly as well, leading to a mean angle error of $0.0039\,\mathrm{rad}$ (see Fig. 6.7).

This section showed impressive results for the control performance of the DSC control law in simulation. In physical robots, however, there are uncertainties in the model parameters as well as unmodeled dynamics. All experiments in this chapter were performed under the assumption of perfectly known system models, as well as negligible friction. These issues are treated in the following chapter.

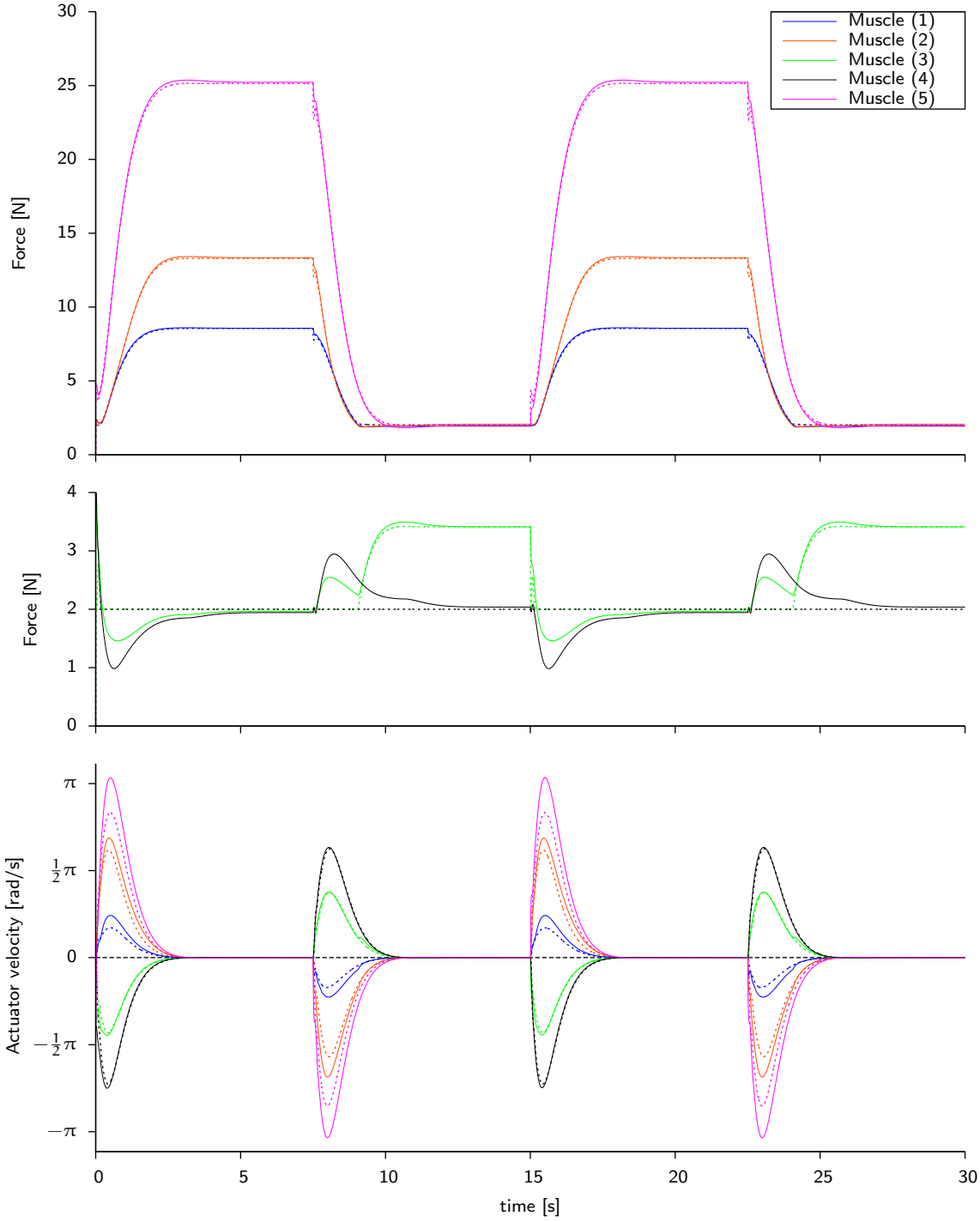


Fig. 6.6: DSC Low-Level Control. The reference (dashed) is shown together with the system output, both for the active forces (top), the inactive forces (center), and the actuator velocity control law (bottom). Forces were controlled to maintain a minimum of 2 N. The trajectory is shown twice for a step up and down.

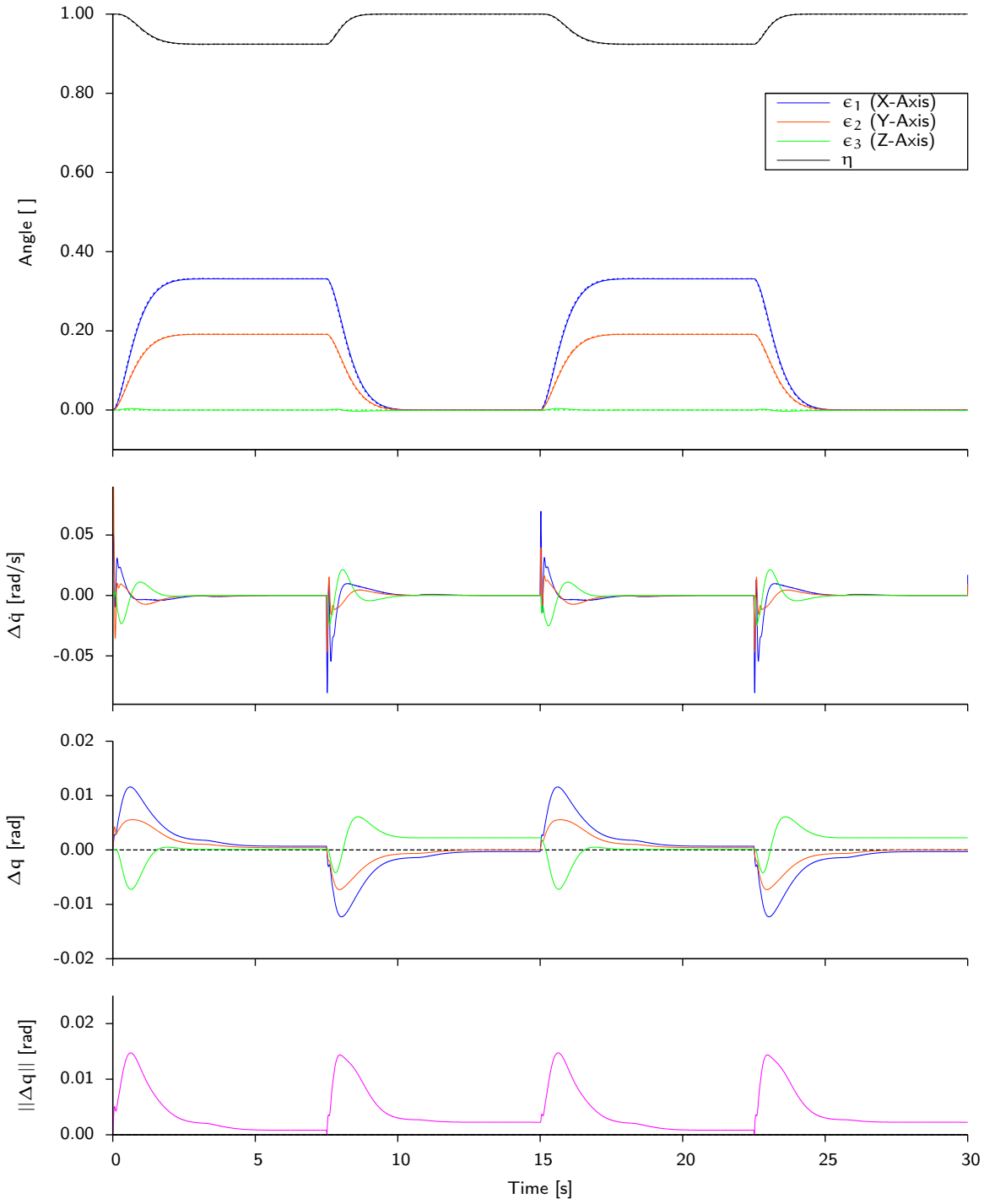


Fig. 6.7: DSC High-Level Control. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in velocity $\Delta\dot{q}$, the error in position Δq, and the absolute angle error (bottom).

7

ADAPTIVE CONTROL

To prove stability of the previously developed controllers, perfectly known system dynamics were assumed. However, due to measurement errors and unmodeled effects this assumption generally does not hold. When this is the case, one of two tools can be used to overcome the problem. Either, the method of robust control or adaptive control is used. The first treats model uncertainties or disturbances at design time so that controllers perform robustly as long as these uncertainties are within well defined bounds. The latter uses *online* adaptation techniques to change control parameters so that after convergence of the adaptive terms, the controller behaves as specified. Which of the two techniques is to be used, depends on the problem. In general it can be said that robustified controllers have faster reaction times on parameter changes, as long as these changes are within the specified bounds. Obviously these bounds have to be known at design time. Adaptive controllers on the other hand can deal with a larger range of parameter deviations and as soon as the adaptation has converged, these controllers are faster [12].

For the problem at hand, the model parameters are comparably well known. However, all previously developed controllers have some unmodeled dynamics, namely friction. Especially, for a spherical joint, where dynamics between the DoF are closely coupled, this can highly degrade performance. Furthermore, due to the dry sliding of the ball on the surface of the socket, these joints exhibit a considerable amount of friction and the lack of high frequency sensory feedback impairs the quality of results for identification procedures. Therefore, the application of adaptive control techniques was chosen to solve the problem of friction compensation.

Adaptive control has been used widely in the field of flexible joint robot control [14, 91, 122, 132]. The problem that was solved was the adaptation of the model parameters, by exploiting the fact that the system dynamics can be represented by a regressor matrix Y and a vector of parameters Θ [119].

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_G(q) = Y(q,\dot{q},\ddot{q})\Theta \tag{7.1}$$

Such a model is called Linear in the Parameters (LP) and can be used in adaptive control, by defining an update law, based on the tracking error. According to Lyapunov's stability theorem, boundedness and even convergence of the model parameters can be proven.

Instead of using a parametric model, it is also possible to use Artificial Neural Networks (ANNs) as a regression model. This can be achieved by choosing the neural network topology so that it can approximate the behavior of the model. The network weights constitute the model parameters. For this purpose Radial Basis Function Networks (RBFNs)

are well suited, as they are linear in the weights (see Section 2.3.3). For instance, this technique has been used in the control of a planar two-link manipulator where the RBFN was used to approximate the system dynamics [30]. A similar technique was used to control a relatively simple tendon-driven mechanism [63], where, however, the muscle Jacobian was not part of the adaptation.

Adaptive control based on RBFNs for robots is quite problematic in terms of scalability, however. In the case of an n DoF robot, the ANN would have a total of $3 \cdot n$ input signals (q, $\dot{q}$, $\ddot{q}$ each $\in \mathbb{R}^n$). Due to the interdependence of all signals, each neuron would then have to consider all inputs, without the possibility of clustering. It is easy to see that the number of neurons, necessary to cover the dynamics, may scale exponentially with n. Therefore, this technique is not suited to learn the full robot dynamics of more complex robots.

Musculosekeletal robots like the Anthrob have been built using accurate manufacturing techniques and the system dynamics can therefore be captured quite accurately from the CAD data. What remains problematic is the identification of useful friction models. Two friction components are relevant for control. These are (1) the friction in the joints and (2) the friction in the muscles due to the routing of the tendons. Therefore friction compensators for the two friction terms are integrated into the Dynamic Surface Control (DSC) controller (see Section 6.2). In the following, first the joint friction, and secondly the muscle friction is evaluated, followed by the development of an adaptive controller.

7.1 JOINT FRICTION

The method of adaptive control has been applied to adapt both to deviations in the dynamic model, as well as a joint friction model for the DLR medical robots with elastic joints [70]. The parametric model chosen for the joint friction compensation consists of Coulomb, load dependent and viscous friction (see Section A.5),

$$\tau_F = (\tau_c + \mu_l|\tau_{act}|) \cdot sgn(\dot{\theta}) + \mu_v \cdot \dot{\theta} = Y_f(\dot{\theta}, \tau_{act})\Theta_f \tag{7.2}$$

where τ_{act} is the actuator torque. The model parameters consist of τ_c which is the static Coulomb friction in the joint torque transmission, μ_l the friction coefficient due to the transmitted load and μ_v the viscous friction coefficient. It can be seen that this model is LP in these parameters which are combined into Θ_f.

Another possibility is again to utilize an RBFN as a friction compensator [31, 33, 40, 87]. Here, instead of learning the dynamics in Eq. 7.1, a friction term similar to Eq. 7.2 was learned. The cross-dependence of the friction terms can be neglected, similar to Eq. 7.2. Therefore, there need to be n neural networks with two input signals each, i. e. the system scales linearly. Furthermore, it has been shown by Ge et al. [33], by direct comparison, that adaptive friction compensators based on RBFNs converge better than the ones based on parametric models.

In the case of a musculoskeletal robot, the definition of the driving load in the joints is more complex, as part of the force is transmitted by the muscles. Hence this dependency

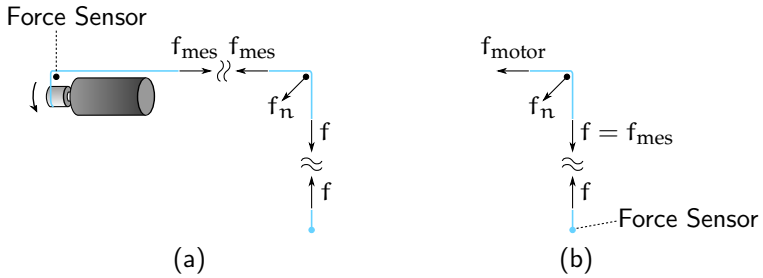


Fig. 7.1: Measuring Tendon Friction. The effect of the friction loss due to the tendon routing mechanism depends on the position of the force sensor. In (a) the force sensor is situated on the driving end, leading to a force measurement that is distorted by friction. In (b) the force sensor is located at the load end, where the force measured is the one actually applied to the attachment point.

is omitted from Eq. 7.2. This leaves us with a model that is solely dependent on the joint velocity $\dot{q}$. The system dynamics equation (see Eq. 6.17) can be redefined to include a joint friction term $\tau_F(\dot{q})$ as follows.

$$H(\alpha)\ddot{q} + C(\alpha, \dot{q})\dot{q} + \tau_G(\alpha) + \tau_F(\dot{q}) = -L^T(\alpha)x_1 \tag{7.3}$$

7.2 MUSCLE FRICTION

Next to the friction in the joints, musculoskeletal robots can exhibit noteworthy friction effects within the transmission system of the tendons. E. g. the ceramic eyelets, used in the Anthrob, deflect the tendons up to 90°, hence causing significant friction losses (see Fig. 7.1). The effects on the control performance depend largely on the position of the muscle force sensor. Generally, there are two schemes, either the force is measured on the driving end (see Fig. 7.1a) or on the load end (see Fig. 7.1b). In the latter case, the force measured is the one actually applied. Here, the low-level force control can deal with the friction with little effort. It is noted, however, that in the case of multi-articular muscles this scheme is not always applicable. The first case features a more compact design which leads to reduced cabling complexity. However, the high-level control has to deal with the tendon friction without being able to effectively measure it. This can be demonstrated by the simulation model used to evaluate the DSC controller (see Section 6.2). Fig. 7.2 depicts the trajectory tracking performance of DSC for the simulation model previously used, with friction parameters increased to match the characteristics that are found in actual systems and force measurement at the driving end. It can be seen that the joint angle follows the reference very poorly. The mean angle error is severely impaired by friction and amounts to $0.246\,\mathrm{rad}$. The friction model necessary to capture all effects would include a Stribeck model, but also the angle at which the eyelet deflects the tendon. Therefore, a model of the following shape could be used.

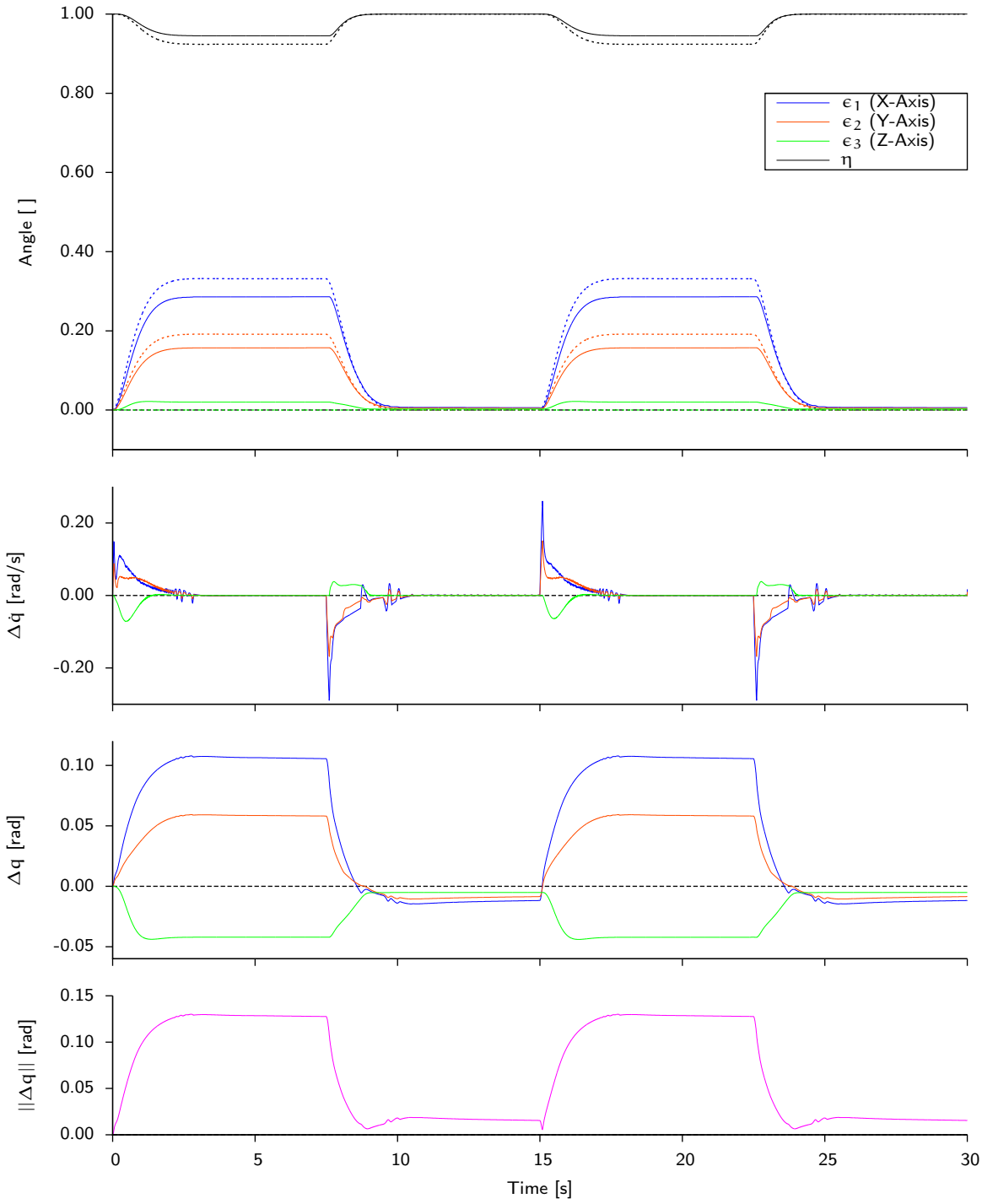


Fig. 7.2: DSC High-Level Control with Friction. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in velocity $\Delta\dot{q}$, the error in position Δq, and the absolute angle error (bottom).

$$\begin{aligned} f_F =& f_c + (f_s - f_c) \cdot e^{-|v(\dot{q})/v_s|^{\delta_s}} + f_v \cdot v(\dot{q}) \\ f_c =& \mu_c \cdot f_n(q, f) \\ f_s =& \mu_s \cdot f_n(q, f) \end{aligned} \tag{7.4}$$

Even though a model of this shape would possibly capture all effects of the tendon friction, the high number of parameters and input signals leaves the adaptation process with a high dimensional search space. This can impede convergence. Instead, one can analyze the effect of each of the input signals on the friction losses, eliminating the ones that have little effect or where the relation is too complex to capture. The largest effect on the tendon friction comes from the transmitted force itself. Furthermore, the angle at which the tendon is deflected depends on all joint angles that the muscle spans which highly increases the number of input signals. For the reason of exponential growth this parameter is omitted. Similarly the dependence on the joint velocity is eliminated and the system dynamics can be extended by a muscle friction term $f_F(x_1)$ (with $x_1 = f$).

$$H(\alpha)\ddot{q} + C(\alpha, \dot{q})\dot{q} + \tau_G(\alpha) = -L^T(\alpha)[x_1 - f_F(x_1)] \tag{7.5}$$

7.3 ADAPTIVE NEURAL NETWORK FRICTION COMPENSATION

To find a controller that compensates both the joint, as well as the muscle friction, the system model is rewritten, utilizing Eq. 7.3 and 7.5.

$$H(\alpha)\ddot{q} + C(\alpha, \dot{q})\dot{q} + \tau_G(\alpha) + \tau_F(\dot{q}) - L^T(\alpha)f_F(x_1) = -L^T(\alpha)x_1 \tag{7.6}$$

Without loss of generality it can be assumed that $\tau_F(\dot{q})$ and $f_F(x_1)$ are bounded function, such that.

$$|\tau_F(\dot{q})| < \underline{\tau}_F \qquad \forall \dot{q} \in \{-\dot{q}_{min}, \dot{q}_{max}\} \tag{7.7}$$

$$|f_F(x_1)| < \underline{f}_F \qquad \forall x_1 \in \{0, f_{max}\} \tag{7.8}$$

Similar to the gravity compensation term τ_G, the two friction terms can be integrated into the controller,

$$\begin{aligned} \phi_1 = -L^{T+}(\alpha)\{&H(\alpha)(\ddot{q}_d + \Lambda_1(\dot{q}_d - \dot{q})) + C(\alpha, \dot{q})(\dot{q}_d + \Lambda_1 \Delta q) \\ &+ \tau_G(\alpha) - K_d r + \hat{\tau}_F(\dot{q})\} + \hat{f}_F(x_1) \end{aligned} \tag{7.9}$$

where $\hat{\tau}_F$ and $\hat{f}_F$ denote estimations of the unknown functions. Due to the fact that all necessary friction components are compensated at this level, the other two levels of control remain unchanged (see Section 6.2). The unknown functions can be approximated by a vector of RBFNs (see Section 2.3.3), where Φ_{JFj} and Φ_{MFj} denote the vector of Radial Basis Functions (RBFs) for the approximation of τ_{Fj} and f_{Fj} respectively,

$$\Phi_{ijk} = \exp - \frac{(x_{ij} - c_{ik})^T \cdot (x_{ij} - c_{ik})}{\sigma_{ik}^2} \qquad i = \{JF, MF\}, j = \{1 \dots o\}, k = \{1 \dots l\} \tag{7.10}$$

where l denotes the number of neurons and o the number of RBFNs per friction compensator which is n for the joint friction and m for the muscle friction. This vector is pre-multiplied with the $l \times 1$ vector of weights $\hat{\Theta}_{ij}$ (see Section 2.3.3).

$$\hat{\tau}_{Fj} = \hat{\Theta}_{JFj}^T \Phi_{JFj} \tag{7.11}$$
$$\hat{f}_{Fj} = \hat{\Theta}_{MFj}^T \Phi_{MFj} \tag{7.12}$$

An adaptation rule for the weights can be found, utilizing the tracking error r and the method of gradient decent [33].

$$\dot{\hat{\Theta}}_{ij} = -\Gamma_{ij} \cdot \Phi_{ij} \cdot r_{ij} \qquad i = \{JF, MF\}, j = \{1 \dots o\} \tag{7.13}$$

where Γ_{ij} is the respective learning factor and r_i is defined as follows.

$$r_{JF} = r \tag{7.14}$$
$$r_{MF} = -L \cdot r \tag{7.15}$$

Adaptation rules have been chosen to exhibit convergence of the model parameters according to Lyapunov's stability theorem, which will be proven in the following section.

7.3.1 *Stability Proof*

The control law in Eq. 7.9 can be rewritten such that,

$$\begin{aligned} \phi_1 = -L^{T+} \{ & H [\ddot{q}_d + \Lambda_1 (\dot{q}_d - \dot{q})] + C [\dot{q}_d + \Lambda_1 \Delta q] \\ & + \tau_G + \tau_F - K_d r - \tilde{\tau}_F \} \\ & + f_F - \tilde{f}_F \end{aligned} \tag{7.16}$$

where $\tilde{\tau}_F$ and $\tilde{f}_F$ represent the respective estimation errors.

$$\tilde{\tau}_F = \tau_F - \hat{\tau}_F \tag{7.17}$$
$$\tilde{f}_F = f_F - \hat{f}_F \tag{7.18}$$

This leads to the closed loop behavior for the skeletal dynamics which, next to the terms from the DSC law, additionally includes the estimation errors.

$$H\dot{r}_1 + Cr + K_d r + \tilde{\tau}_F - L^T \tilde{f}_F = -L^T [z_1 - \mu_1 \dot{s}_1] \tag{7.19}$$

It has been shown that an ANN of the given form can approximate any continuous function, given that the number of neurons is sufficient and the parameters of the RBFs are appropriate [104], up to a bounded error $\epsilon_{JF} \in \mathbb{R}^n$ and $\epsilon_{MF} \in \mathbb{R}^m$, respectively. Therefore, each friction term can be written as

$$\mathcal{F}_{ij} = \Theta_{ij}^T \Phi_{ij} + \epsilon_{ij} \qquad i = \{JF, MF\}, j = \{1 \dots o\} \tag{7.20}$$

where $\mathcal{F}_i$ denotes the joint friction and the muscle friction, respectively and $|\epsilon_i| \leqslant \underline{\epsilon}_i$. In the following, it is assumed that the parameters of the neural network have been chosen accordingly. Therefore $\tilde{\Theta}_{ij}$ can be defined to denote the error in the weights as follows.

$$\tilde{\mathcal{F}}_{ij} = \Theta_{ij}^T \Phi_{ij} + \epsilon_{ij} - \hat{\Theta}_{ij}^T \Phi_{ij} = (\Theta_{ij}^T - \hat{\Theta}_{ij}^T)\Phi_{ij} + \epsilon_{ij} = \tilde{\Theta}_{ij}^T \Phi_{ij} + \epsilon_{ij} \tag{7.21}$$

A Lyapunov function candidate is found by extending the previous one. Two terms are added for each of the errors in the weights

$$\begin{aligned} V = &+ \frac{1}{2} r^T H r + \frac{1}{2} z_1^T z_1 + \frac{1}{2} z_2^T z_2 + \frac{k_e}{2} e_1^T e_1 + \frac{k_e}{2} e_2^T e_2 \\ &+ \frac{1}{2} \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \tilde{\Theta}_{JFj} + \frac{1}{2} \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \tilde{\Theta}_{MFj} \end{aligned} \tag{7.22}$$

The derivative of the Lyapunov function can be stated as follows.

$$\begin{aligned} \dot{V} = &- r^T K_d r - z_1^T \Lambda_2 z_1 - z_2^T \Lambda_3 z_2 \\ &+ r^T L^T \mu_1 \dot{s}_1 - z_1^T g_2 \mu_2 \dot{s}_2 + k_e e_1^T \dot{e}_1 + k_e e_2^T \dot{e}_2 \\ &- r^T \tilde{\tau}_F + r^T L^T \tilde{f}_F + \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \dot{\tilde{\Theta}}_{JFj} + \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \dot{\tilde{\Theta}}_{MFj} \\ = &+ \dot{V}_{bs} + \dot{V}_{dsc} + \dot{V}_{ad} \end{aligned} \tag{7.23}$$

Both $\dot{V}_{bs}$ and $\dot{V}_{dsc}$ have been dealt with in the previous chapter (see Section 6.1 and Section 6.2, respectively). Therefore only $\dot{V}_{ad}$ is evaluated here. Assuming that the RBFNs can fully approximate the friction terms up to the bounded error ϵ_i, leaves the weights Θ_{ij} to be constant. Hence, we note that $\dot{\Theta}_{ij}$ is zero and state the following.

$$\dot{\tilde{\Theta}}_{ij} = \dot{\Theta}_{ij} - \dot{\hat{\Theta}}_{ij} = -\dot{\hat{\Theta}}_{ij} \tag{7.24}$$

Introducing this and the learning rule from Eq. 7.13, $\dot{V}_{ad}$ can be rewritten. Subsequently, Eq. 7.21 is inserted.

$$\begin{aligned} \dot{V}_{ad} &= -r^T \tilde{\tau}_F + r^T L^T \tilde{f}_F + \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \dot{\tilde{\Theta}}_{JFj} + \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \dot{\tilde{\Theta}}_{MFj} \\ &= -r^T \tilde{\tau}_F + r^T L^T \tilde{f}_F + \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Phi_{JFj} r_j - \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Phi_{MFj} (Lr)_j \\ &= -r^T \tilde{\tau}_F + r^T L^T \tilde{f}_F + \sum_{j=1}^{n} \tilde{\tau}_{Fj} r_j - \sum_{j=1}^{m} \tilde{f}_{Fj} (Lr)_j - \epsilon_{JF}^T r + \epsilon_{MF}^T Lr \\ &\leqslant \underline{\epsilon}_{JF}^T |r| + \underline{\epsilon}_{MF}^T |Lr| \end{aligned} \tag{7.25}$$

Hence, the Lyapunov function derivative can be written as follows.

$$\dot{V} \leqslant \dot{V}_{bs} + \dot{V}_{dsc} + \underline{\epsilon}_{JF}^T |r| + \underline{\epsilon}_{MF}^T |Lr| \tag{7.26}$$

Even though Uniformly Ultimately Bounded (UUB) stability is hereby shown for the adaptive controller, the proof of stability does not place a bound on the adaptive parameters. Due to the fact that there is a non-linear error ϵ_i which cannot be approximated by the ANN, the network weights $\hat{\Theta}_i$ may grow indefinitely [48]. The proposed solution is to limit the gradient in some way, e. g. by introducing the so called σ-Modification into the learning rule [32],

$$\dot{\hat{\Theta}}_{ij} = -\Gamma_{ij} \cdot \Phi_{ij} \cdot r_{ij}^T - \sigma_i \cdot \hat{\Theta}_{ij} \tag{7.27}$$

which can be seen as a damping term for the network weights, where σ_i is the positive damping constant. This modification of the parameter update rule leads to a change in the Lyapunov function derivative. Again we only need to consider the part of $\dot{V}$ which was introduced by the adaptive terms (see Eq. 7.25).

$$\begin{aligned}
\dot{V}_{ad} =& - r^T \tilde{\tau}_F + r^T L^T \tilde{f}_F + \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \dot{\tilde{\Theta}}_{JFj} + \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \dot{\tilde{\Theta}}_{MFj} \\
=& - r^T \tilde{\tau}_F + r^T L^T \tilde{f}_F + \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Phi_{JFj} r_j - \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Phi_{MFj} (Lr)_j \\
&+ \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \sigma_{JF} \hat{\Theta}_{JFj} + \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \sigma_{MF} \hat{\Theta}_{MFj} \\
=& - \epsilon_{JF}^T r + \epsilon_{MF}^T Lr \\
&+ \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \sigma_{JF} \left[\Theta_{JFj} - \tilde{\Theta}_{JFj} \right] \\
&+ \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \sigma_{MF} \left[\Theta_{MFj} - \tilde{\Theta}_{MFj} \right]
\end{aligned} \tag{7.28}$$

Finally, by employing Young's inequality (see Eq. 6.59), an upper bound can be placed on $\dot{V}$,

$$\begin{aligned}
\dot{V} \leqslant & + \dot{V}_{bs} + \dot{V}_{dsc} + \underline{\epsilon}_{JF}^T |r| + \underline{\epsilon}_{MF}^T |Lr| \\
& - \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \Gamma_{JFj}^{-1} \sigma_{JF} \tilde{\Theta}_{JFj} - \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \Gamma_{MFj}^{-1} \sigma_{MF} \tilde{\Theta}_{MFj} \\
& + \frac{1}{4c_5} \sum_{j=1}^{n} \tilde{\Theta}_{JFj}^T \tilde{\Theta}_{JFj} + \frac{1}{4c_6} \sum_{j=1}^{m} \tilde{\Theta}_{MFj}^T \tilde{\Theta}_{MFj} \\
& + c_5 \sum_{j=1}^{n} \Theta_{JFj}^T \sigma_{JF}^2 \Gamma_{JFj}^{-2} \Theta_{JFj} + c_6 \sum_{j=1}^{m} \Theta_{MFj}^T \sigma_{MF}^2 \Gamma_{MFj}^{-2} \Theta_{MFj}
\end{aligned} \tag{7.29}$$

where $c_{5/6}$ can be chosen arbitrarily. Due to the fact that the DSC controller has been shown to be UUB, and both r and Lr are bounded, the same is proven for the adaptive

control law, according to Lyapunov's stability theorem, for the control gains given in Section 6.2.2.

Another possibility for placing a bound on the network weights $\hat{\Theta}_{ij}$ is to turn off the learning when a certain maximum value has been reached [40].

$$\dot{\hat{\Theta}}_{ij} = \begin{cases} 0 & \text{if} \quad |\hat{\mathcal{F}}_{ij}| > \mathcal{F}_{ijmax} \quad \text{and} \quad sgn(-\Gamma_{ij} \cdot \Phi_{ij} \cdot r_{ij}) = -sgn(\hat{\mathcal{F}}_{ij}) \\ -\Gamma_{ij} \cdot \Phi_{ij} \cdot r_{ij} & \text{otherwise} \end{cases} \tag{7.30}$$

In the following, a third possibility was chosen, which is to to use the σ-Modification only when the approximated function is outside of physically consistent bounds [32].

$$\dot{\hat{\Theta}}_{ij} = \begin{cases} -\Gamma_{ij} \cdot \Phi_{ij} \cdot r_i^T - \sigma_i \cdot \hat{\Theta}_{ij} & \text{if} \quad \hat{\mathcal{F}}_{ij} < \mathcal{F}_{ijmin} \\ -\Gamma_{ij} \cdot \Phi_{ij} \cdot r_i^T & \text{if} \quad \mathcal{F}_{ijmin} \leqslant \hat{\mathcal{F}}_{ij} \leqslant \mathcal{F}_{ijmax} \\ -\Gamma_{ij} \cdot \Phi_{ij} \cdot r_i^T - \sigma_i \cdot \hat{\Theta}_{ij} & \text{if} \quad \hat{\mathcal{F}}_{ij} > \mathcal{F}_{ijmax} \end{cases} \tag{7.31}$$

This has the advantage that weights are free to adapt to changes in the friction terms, as long as they are within a physically consistent range. Outside of this range, the weights are damped by the σ-modification. Stability for this learning rule can be easily proven by a combination of the two Lyapunov function derivatives in Eq. 7.25 and Eq. 7.29, while the former is employed for weights inside the bounds given by $\mathcal{F}_{imin}$ and $\mathcal{F}_{imax}$ and the latter outside.

7.3.2 *Controller Discussion*

The integration of adaptive friction compensation terms changes only the high-level control law. Therefore, the low-level controllers can still be easily implemented on distributed nodes without additional performance concerns, as the neural network approximation is only performed on the central control unit.

While for the joint friction symmetric bounds τ_{Fmin} and τ_{Fmax} were chosen, the muscle friction compensation utilizes Eq. 7.31 to ensure muscle frictions that are always positive which is physically consistent. Here, the difficulty lies in the fact that trajectory errors are in $\mathbb{R}^n$, while the muscle friction is $\mathbb{R}^m$. Therefore, the learning has a null-space where adaptation cannot be controlled by the trajectory tracking error r. However, this is the only error measure available, since the muscle force at the load side is generally not measurable (see Fig. 7.1). Therefore, the null-space muscle friction is set to zero, to avoid co-contraction, by pre-multiplying the RBFN output by the muscle Jacobian and subsequently utilizing the optimization procedure to obtain the reference forces.

$$\begin{aligned}\Phi_1 = -L^{T+}(\alpha)\{&H(\alpha)\left[\ddot{q}_d + \Lambda_1(\dot{q}_d - \dot{q})\right] + C(\alpha,\dot{q})\left[\dot{q}_d + \Lambda_1 \Delta q\right] \\ &+ \tau_G(\alpha) - K_d\left[\dot{q} - \dot{q}_d - \Lambda_1 \Delta q\right] \\ &+ \hat{\tau}_F - L^T \hat{f}_F\}\end{aligned}$$

$$\Phi_2 = g_1^{-1}(f)\left\{\frac{\Phi_1 - s_1}{\mu_1} + L(\alpha)\left[\dot{q} - \dot{q}_d - \Lambda_1 \Delta q\right] - \Lambda_2\left[f - s_1\right] - f_1(q,\dot{q},f)\right\}$$

$$\Phi_3 = g_2^{-1}\left\{\frac{\Phi_2 - s_2}{\mu_2} - g_1(f)\left[f - s_1\right] - \Lambda_3\left[\dot{\theta} - s_2\right] - f_2(f,\dot{\theta})\right\}$$

To reduce the complexity of the learning task it was further assumed that there is no cross coupling between the friction terms. This led to the fact that each element of a friction vector could be adapted independently. Therefore, instead of having n inputs to the joint friction and m to the muscle friction RBFN, there were n and m neural networks with one input, each, for the muscle and joint friction adaptation, respectively. It is obvious that this simplification majorly reduces the parameter as well as the learning space for the RBFNs. Similar to other ANNs it is useful to normalize the inputs to the neural networks $\dot{q}$, and f to ± 1. This is especially helpful for network parameterization, as centers of the RBFs can be chosen to match the normalized input and are not dependent on range of the actual input signals.

For the resulting adaptive control law, simulation results were obtained, similarly to the previous basic *backstepping* and DSC controllers. To demonstrate the adaptive compensators, friction parameters in the Stribeck model for the joint friction and the Coulomb model for the muscle friction were increased by a factor of 10. Parameters for the RBFN compensators were chosen to match the functions, leading to 11 neurons with equidistant RBF centers for the muscle friction and 15 neurons with centers spaced logarithmically for higher density around zero. The latter allows for capturing the non-linear behavior of the Stribeck model which is not present in the Coulomb model. It can be seen that trajectory tracking errors, even in the presence of the higher friction losses, are highly reduced. Furthermore, the tracking performance increases from the first to the second step, due to the learning. The mean angle error amounts to $0.011\,\mathrm{rad}$ and $0.009\,\mathrm{rad}$, respectively.

A comparison of the mean angle errors was performed for all developed *backstepping* controllers. This evaluation shows the performance of the previously developed controllers for the low friction case (see Fig. 7.4a), followed by the high friction case, where the muscle force was measured at the driving end (see also Fig. 7.1). The tracking error, in the latter case, was not significantly increased by the higher friction parameters, since the low-level controller was able to compensate for parts of the muscle friction (see Fig. 7.4b).

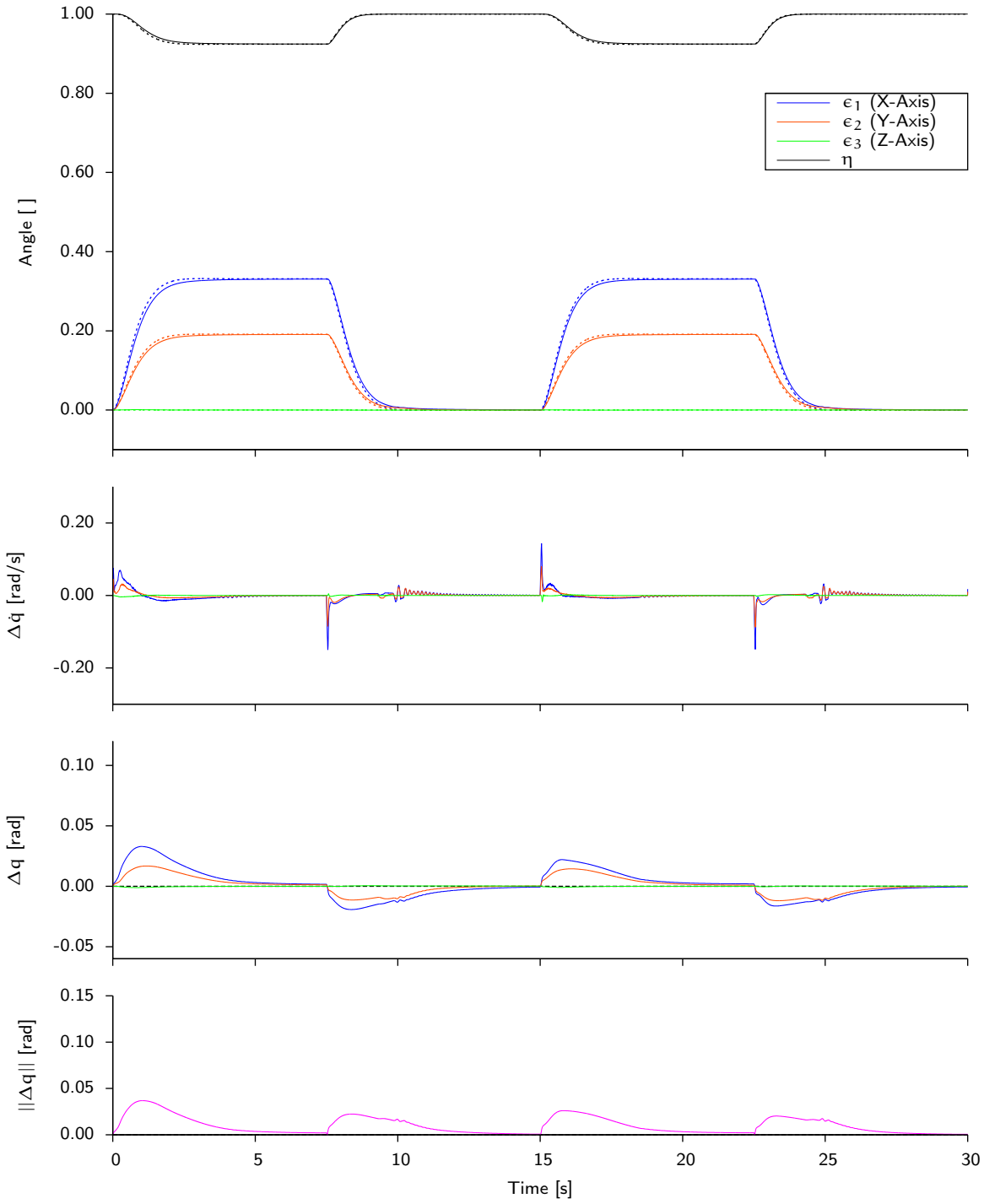


Fig. 7.3: Adaptive Friction Compensation. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in velocity $\Delta\dot{q}$, the error in position Δq, and the absolute angle error (bottom).

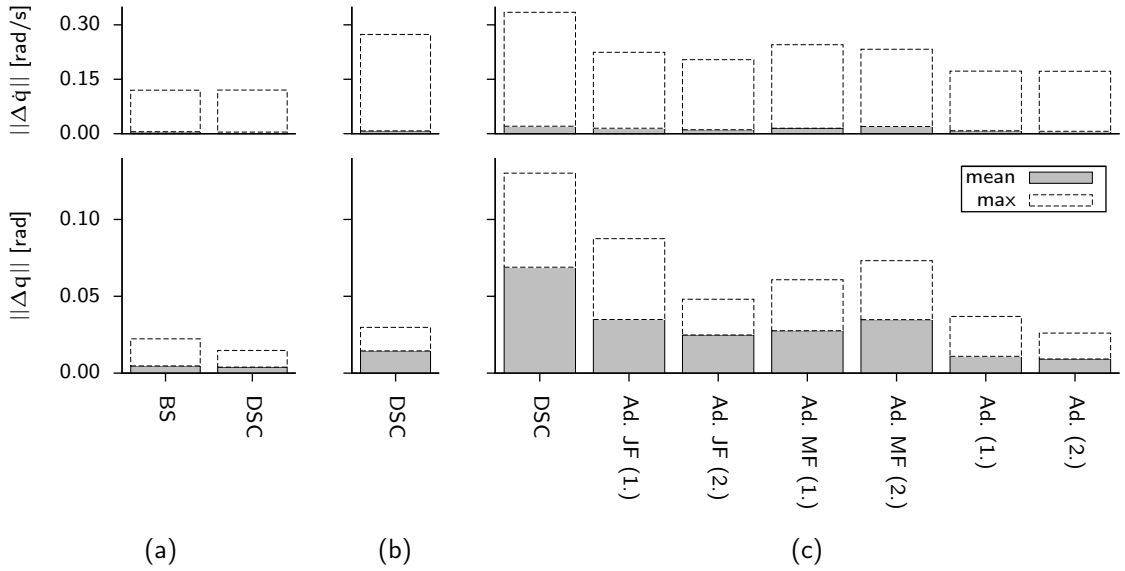


Fig. 7.4: Adaptive Controller Comparison. Joint velocity errors $\|\Delta\dot{q}\|$ and joint angle errors $\|\Delta q\|$ are depicted, both for mean and maximum values. For reference, (a) shows the low friction experiments from Chapter 6 for the backstepping (BS) and the DSC control law, while (b) shows the DSC control law for high joint and muscle friction, using the force measurement at the driving end. Finally, in (c) experiments with high joint and muscle friction and force measurement at the load end are depicted for the DSC controller and the adaptive (Ad.) controllers with only joint (JF), only muscle (MF) and finally both friction compensation terms. The behavior of the adaptive terms over time is depicted by showing two consecutive executions of the trajectory.

However, when forces were measured at the load end (see Fig. 7.4c), the performance degraded massively for the DSC controller, which was improved by the introduction of joint or muscle friction compensation It can be seen that if solely muscle friction compensation was used, the learning was actually diverging from one step to the next, which was due to the fact that the muscle friction compensator was trying to learn both frictions at the same time. The application of both friction compensators performed best, while a learning from the first to the second step was visible, leading to the conclusion that the learning was converging.

8

RESULTS

The results presented in this chapter give evidence of the trajectory tracking performance of the controllers developed in Chapter 5, Chapter 6 and Chapter 7. A comparison between the different techniques is presented, utilizing the trajectory tracking error as a benchmark. As a physical platform, the musculoskeletal robot arm Anthrob (see Chapter 3) was chosen, performing a stepwise evaluation. First, the spherical shoulder joint with the eight muscles of the rotator cuff was considered, subsequently moving on to the full arm, including the elbow joint and the bi-articular Biceps. Hence results were obtained for spherical joint control with colliding muscles, as well as for controlling systems with bi-articular muscles. Joint angles were measured by an analog potentiometer in the elbow joint and a stereo camera system with infrared markers on the fixed scapula and the humerus to obtain the state of the spherical shoulder joint. Even though this motion capture system features comparably low latencies of 4.3 ms in average [139], the frame rate was limited to 60 fps which complicated the numerical differentiation of the joint angle to obtain the joint velocities. For this purpose a high-gain observer [60] was utilized to improve the quality of both the angular position measurements, as well as the angular velocities.

A high-gain observer utilizes a non-linear system model to estimate all system states, based on system inputs and measurements of the system output. For musculoskeletal robots the non-linear system model has been derived in Chapter 4, such that the relationship between system inputs and outputs is well known. The system states α and $\dot{q}$ are redefined to x_1 and x_2, respectively, forming the system model for the high-gain observer as follows (see also Eq. 4.16),

$$\begin{aligned} \dot{x}_1 &= A(x_1)x_2 \\ \dot{x}_2 &= \gamma(x_1, x_2, f) \end{aligned} \tag{8.1}$$

where γ denotes the full system dynamics. As the camera system measures the angular position, the measurement y is equal to α and the high-gain observer can be defined as follows to serve as an estimator for α and $\dot{q}$ [60],

$$\begin{aligned} \hat{\dot{x}}_1 &= A_1(\hat{x}_1)\,[\hat{x}_2 + h_1 \cdot \Delta q] \\ \hat{\dot{x}}_2 &= \gamma_0(x_1, x_2, f) + h_2 \cdot \Delta q \end{aligned} \tag{8.2}$$

where $\hat{x}_{1/2}$ denote estimates of the respective system states and γ_0 the nominal system model. Here, the angle error Δq according to Eq. 5.23 is utilized with the positive definite gains h_1 and h_2 to estimate the system states. Since the controller utilizes a model of

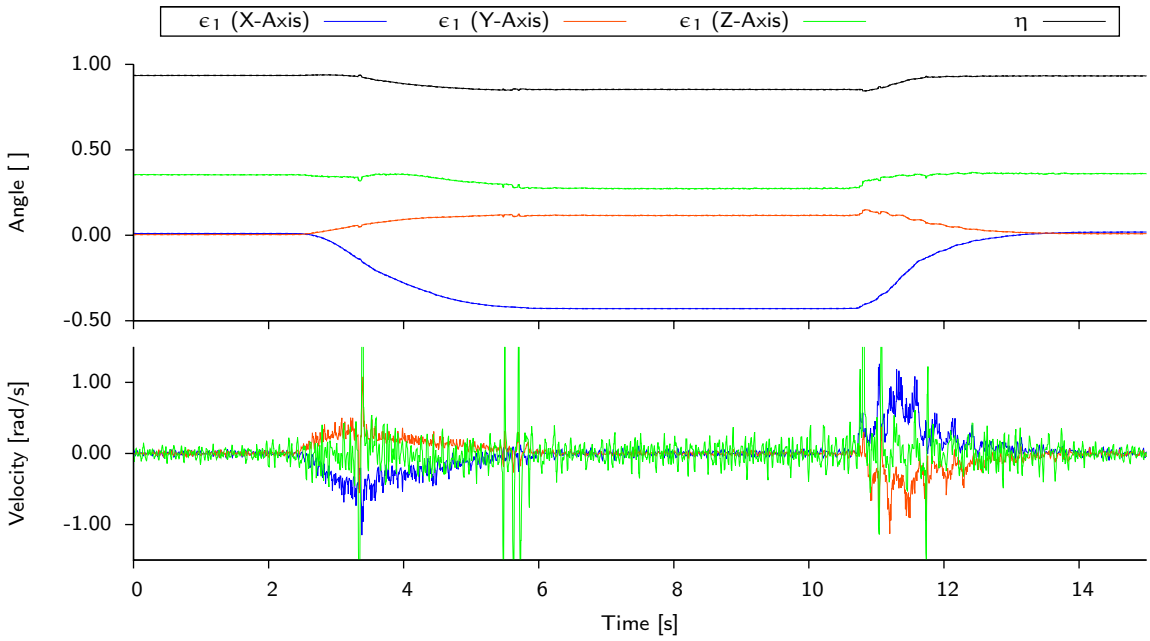


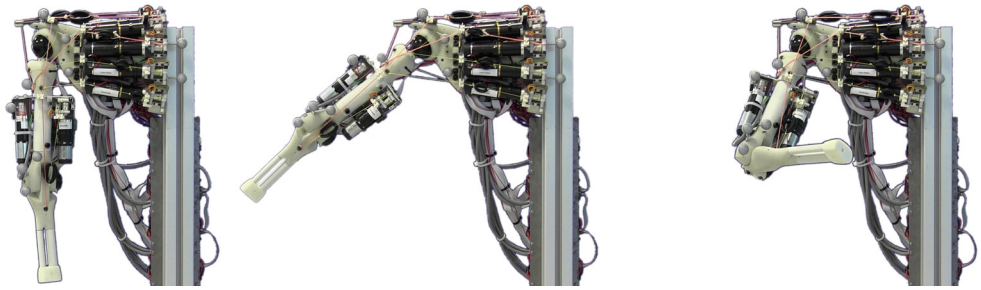
Fig. 8.1: High-Gain Observer for Shoulder Joint. The angle is shown (top), both for the motion capture data, as well as the filtered angle (dashed), along with the estimated velocity (bottom).

the system dynamics to compensate for any non-linearities, the result of γ_0 is essentially equal to the reference acceleration.

Estimates for the shoulder joint state can be seen in Fig. 8.1. Although, joint angles could only be measured at a frequency of 60 Hz, useful estimates of the angular velocity were obtained from the high-gain observer. The joint angle measurement itself was improved by the filter as well.

8.1 SPHERICAL JOINT CONTROL

The developed controllers were evaluated for the control of the spherical shoulder joint with dismounted forearm. A reference trajectory was obtained by choosing three poses, subsequently utilizing Eq. 5.39 to interpolate. The resulting trajectory was therefore described by reference accelerations, velocities and angles. The three different poses (see Fig. 8.2) are (1) the resting position, which presents the shoulder adduction, (2) an abduction (extended to the side) of $0.9\,\mathrm{rad}$ ($\sim 51.6°$), and (3) an anteversion (drawn to the front) of $0.67\,\mathrm{rad}$ ($\sim 38.4°$). The zero position formed a pose where the elbow axis was rotated inwards by 45°, similar to the relaxed position of a human arm . This trajectory was chosen, as it presents substantial coverage of the arm's movement capabilities. It was executed in several experiments by the control laws obtained by the techniques of Computed Force Control (CFC) and Dynamic Surface Control (DSC) and finally the dif-



(a) (b) (c)

Fig. 8.2: Anthrob Trajectory. The trajectory is generated by interpolation between three poses, which are (a) the shoulder adduction, (b) the shoulder abduction and (c) the shoulder anteversion. For the full Anthrob the elbow joint performs a flexion between the shoulder abduction and the adduction.

ferent combinations of adaptive controllers. A compilation of the control parameters can be found in Table B.6.

8.1.1 *Computed Force Control*

The first controller that was evaluated was the CFC control law developed in Chapter 5. It was utilized as a trajectory tracking controller to be comparable to the controllers based on the *backstepping* method. Even though this control law performed well for simpler systems with revolute joints (see Section 5.3 and [23, 58, 72, 113]), it can be seen that the closely coupled dynamics of several muscles spanning a more complex joint, led to oscillations in the angle during transitions between poses (see Fig. 8.4 on page 120). These oscillations were visible during the transition from the abduction to the anteversion ($\sim 5\,\mathrm{s}$ to $7\,\mathrm{s}$) and from the anteversion to the adduction ($\sim 10\,\mathrm{s}$ to $12\,\mathrm{s}$). It is obvious that this is a consequence of neglecting the low-level control dynamics. The different response times of the different muscles, led to movements orthogonal to the reference trajectory, consequence of which were the apparent oscillations in the angle. Furthermore, trajectory tracking could only be achieved with large errors, even for the three steady states (at 5, 10 and 15 s) The largest steady state error was apparent for the abduction and can be stated as $0.29\,\mathrm{rad}$ ($\sim 16.8°$), while the mean trajectory tracking error over the whole trajectory was $0.24\,\mathrm{rad}$ ($\sim 13.8°$). For a full account of the tracking errors for the whole set of experiments refer to Table B.7.

8.1.2 *Dynamic Surface Control*

Both steady state as well as overall tracking errors were reduced by the novel DSC control law, where the oscillations could be fully eliminated (see Fig. 8.5 on page 121). This was

due to the systematic approach of obtaining a set of control laws for the complete system dynamics, without neglecting low-level dynamics. Steady state errors for the abduction and the adduction were largely reduced (by approximately a factor of 2.5). The largest steady state offset was apparent for the anteversion, which was slightly larger than for the CFC control law.

The main difficulty in the design of the actuation for a spherical joint lies in anticipating the exact movement capabilities beforehand. Due to the fact that muscle lever arms change with the pose, there may be poses which can only be reached by large efforts of individual muscles. In these poses there are simply no muscles which can effectively pull the arm in the desired direction. Obviously this flaw could be addressed by changes in the design of the actuation system, but in the case of the Anthrob, placement of the muscle insertion point was inspired by anatomic considerations. Therefore, it can be concluded that limitations in the movement capabilities were a result of the design simplifications. It is clear that the anteversion of a human arm is not a pure glenohumeral movement but is to a large degree facilitated by a rotational movement of the scapula. The immobilization of the scapula in the current robot design hence explains larger steady state offsets for the anteversion. Even though it is possible to reduce the errors in this pose, e. g. by larger gains or by the adaptive techniques discussed in this work, comparably large errors will persist due to the chosen actuator placement.

8.1.3 *Adaptive Control*

Due to the effects of friction, both in the joint, as well as in the tendon routing mechanism, the utilization of adaptive friction compensation terms can improve the control performance (see Chapter 7). The effects of the two developed adaptive terms was evaluated by performing three experiments, starting with (1) the joint friction compensation (see Fig. 8.6 on page 122), to (2) the muscle friction compensation (see Fig. 8.7 on page 123) and finally going to (3) the combination of both (see Fig. 8.8 on page 124). The overall tracking error was reduced for any of the three experiments, while the combination of both friction compensators performed best. Here, the steady state offset for the abduction could be reduced to $0.07\,\mathrm{rad}$ ($\sim 4.1°$). The highest steady state error remains for the anteversion which amounts to $0.12\,\mathrm{rad}$ ($\sim 6.8°$). A comparison, displaying the tracking errors for the different control laws, is shown in Fig. 8.3. It can be seen that both maximum as well as mean tracking errors could be highly reduced by utilizing DSC and even further by the adaptive friction terms. By executing the trajectory several times, convergence of the adaptive terms was shown. For any of the adaptive control experiments, the mean absolute trajectory tracking error was reduced for the second iteration.

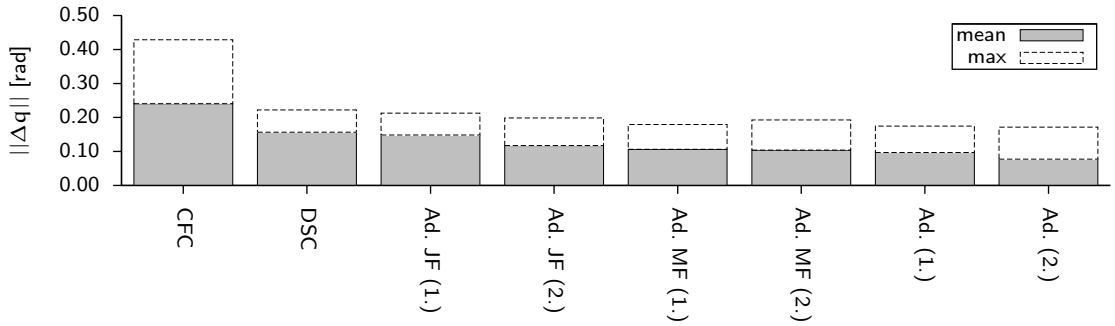


Fig. 8.3: Controller Comparison for Anthrob Shoulder Joint. Mean and maximum absolute joint angle errors are shown for different controllers: (1) CFC, (2) DSC, and combinations (3-5) of the adaptive terms for joint friction (JF) and muscle friction (MF). Behavior of adaptive terms is depicted by showing errors of two consecutive executions.

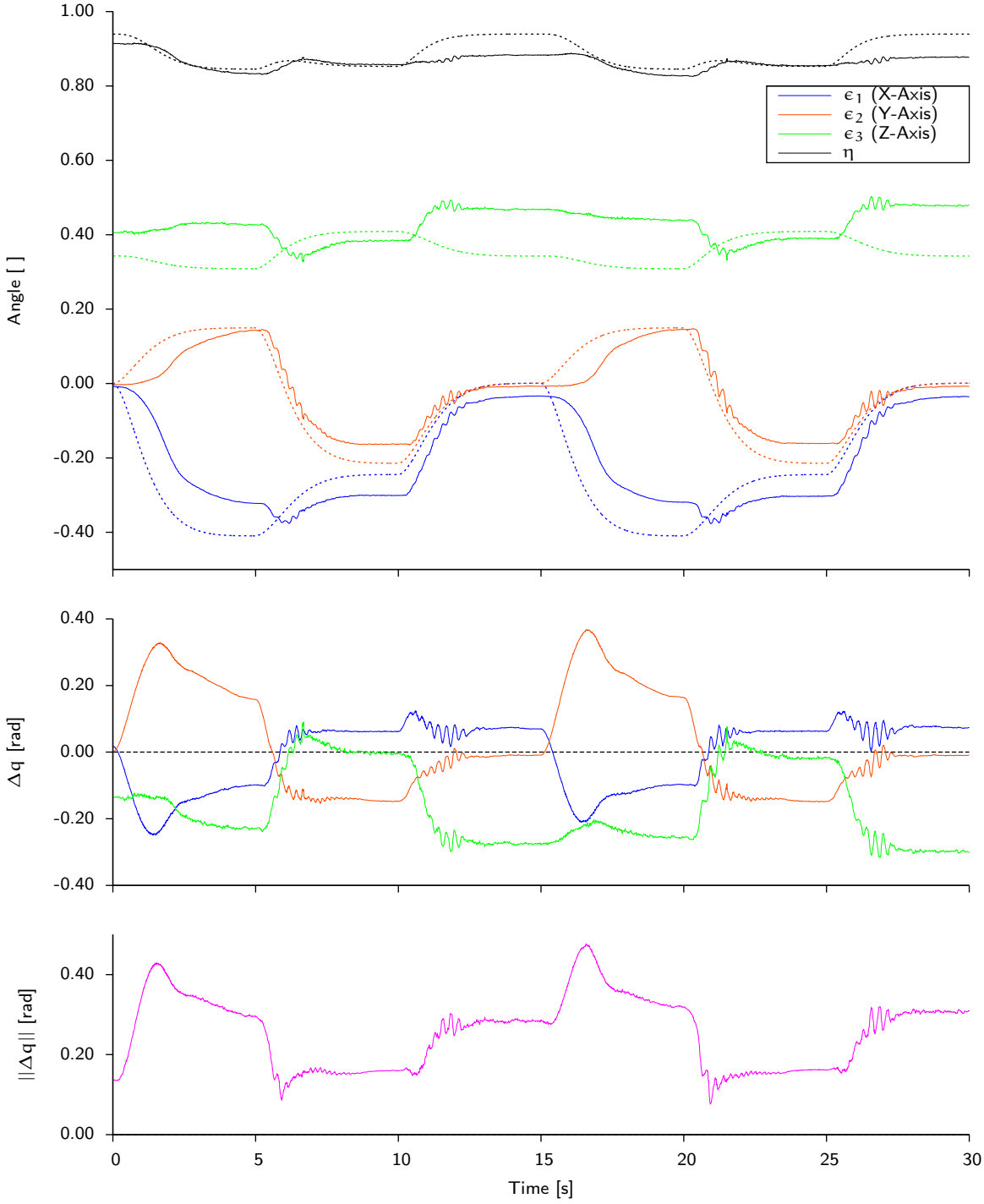


Fig. 8.4: CFC for Anthrob Shoulder Joint. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in position Δq (center) and the absolute angle error (bottom).

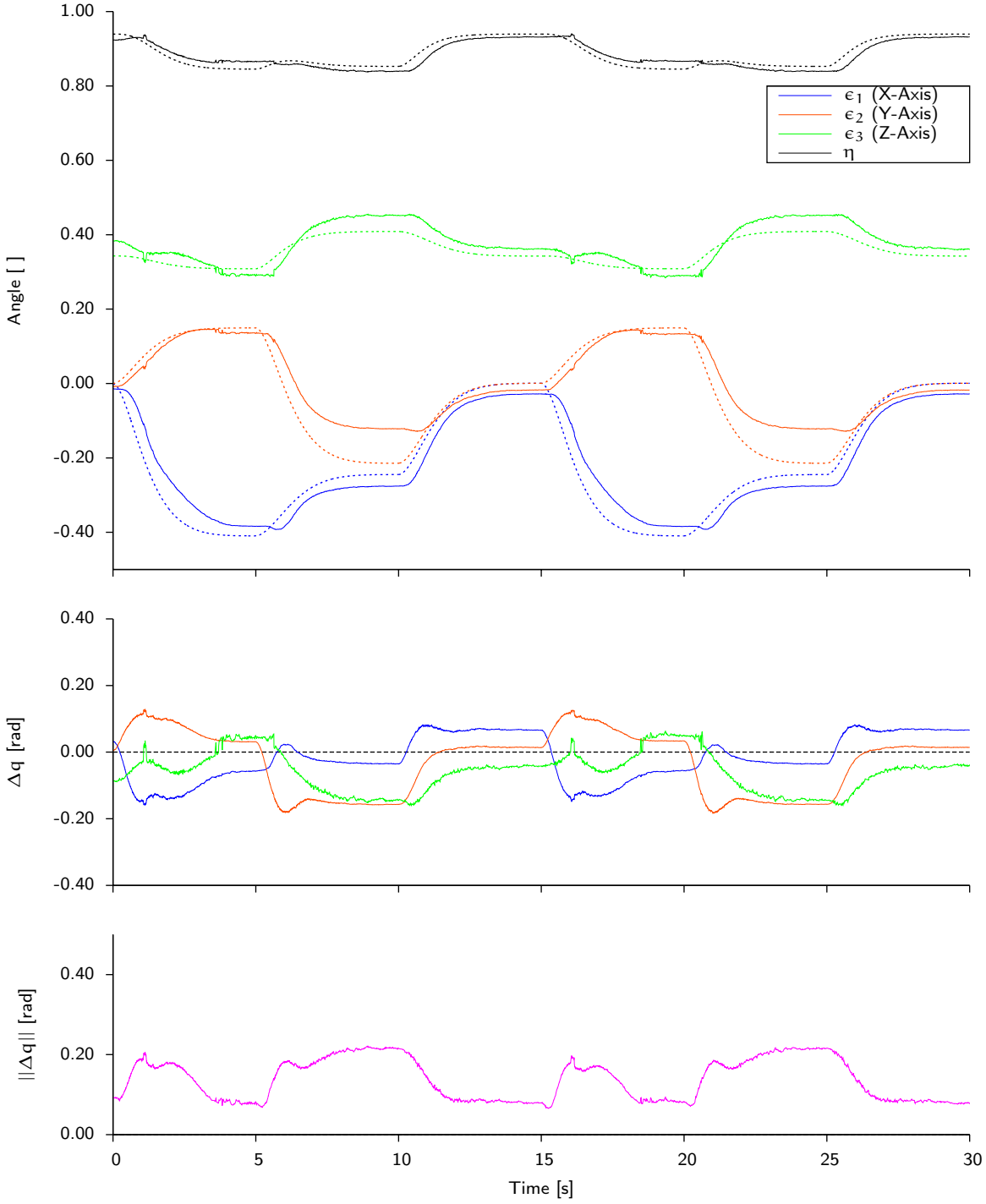


Fig. 8.5: DSC for Anthrob Shoulder Joint. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in position Δq (center) and the absolute angle error (bottom).

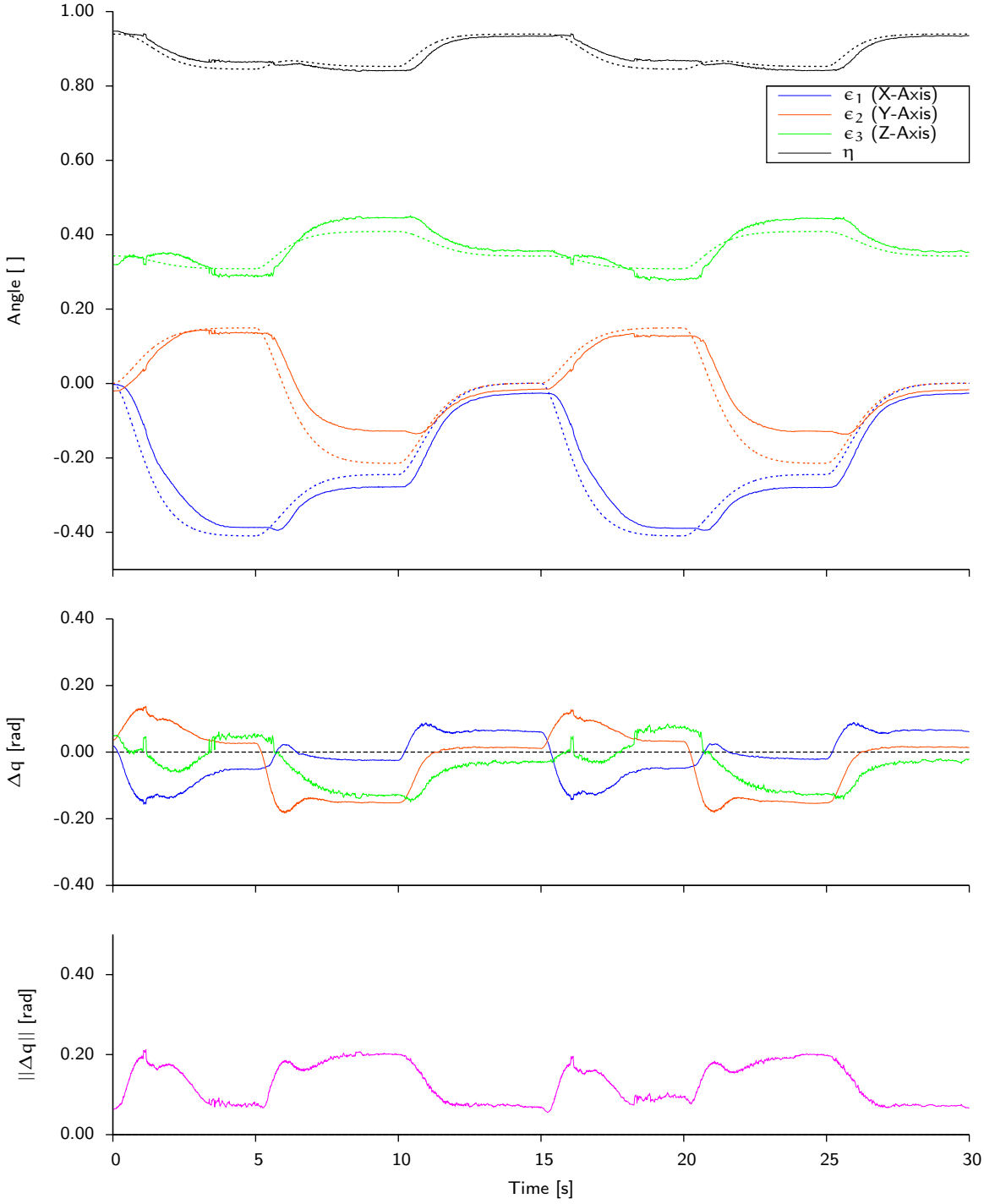


Fig. 8.6: Joint Friction Compensation for Anthrob Shoulder Joint. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in position Δq (center) and the absolute angle error (bottom).

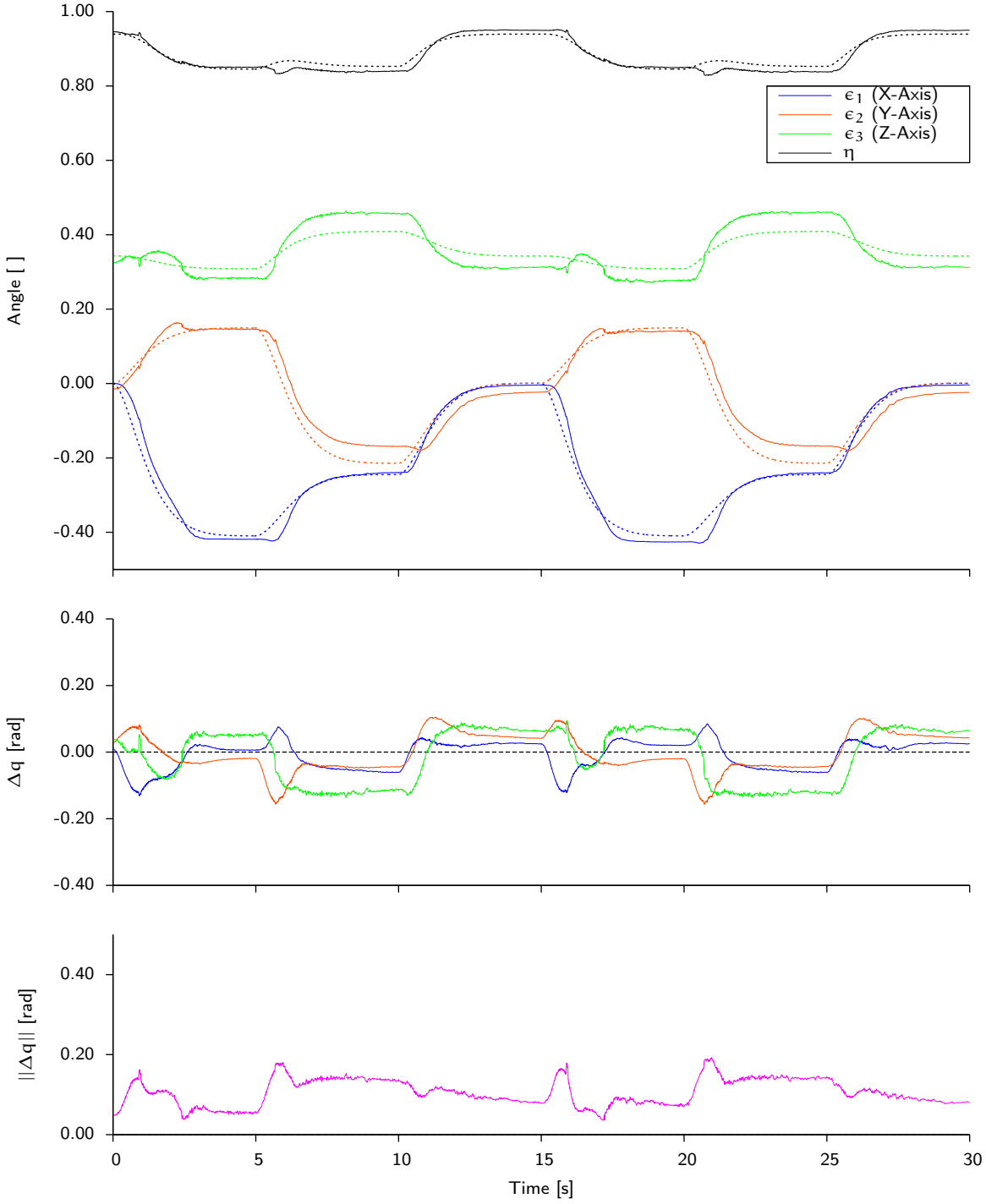


Fig. 8.7: Muscle Friction Compensation for Anthrob Shoulder Joint. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in position Δq (center) and the absolute angle error (bottom).

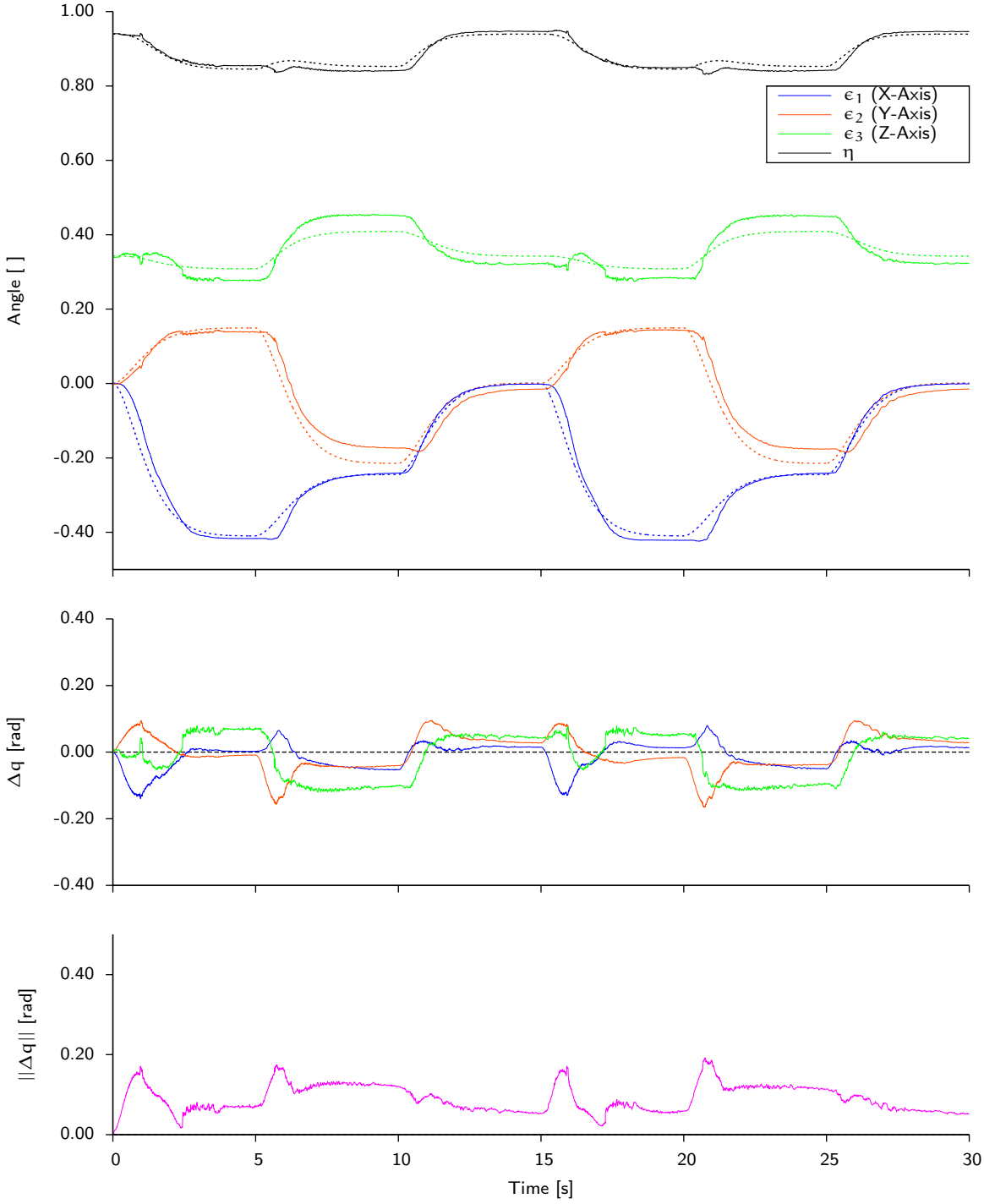


Fig. 8.8: Joint & Muscle Friction Compensation for Anthrob Shoulder Joint. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the error in position Δq (center) and the absolute angle error (bottom).

8.2 ANTHROB CONTROL

Both the DSC, as well as the adaptive control laws were evaluated for the control of the full Anthrob, including the elbow joint and the bi-articular Biceps. Here, the same trajectory as for the shoulder was utilized, while an elbow flexion was executed during the transition between the abduction and the anteversion (see Fig. 8.2). The resting pose of the elbow was set to $0.2\,\mathrm{rad}$ ($\sim 11.45°$) and the flexed pose to $0.5\,\pi$ ($90°$). The elbow extension was performed during the shoulder adduction (see Fig. 8.10 on page 126). A full account of the control parameters is given in Table B.8. It can be seen that good tracking performance could be achieved for both joints, with similar performance as for the pure shoulder control and negligible tracking and steady state errors for the elbow joint. A comparison between the non-adaptive and the whole range of previously shown adaptive controllers was performed, giving evidence that the adaptive friction compensation terms were again capable of improving the control performance (see Fig. 8.9).

Even though, final control performance of the shoulder is comparable to the results obtained for the shoulder alone, the inclusion of the bi-articular Biceps led to a very different muscle force distribution. During the anteversion, the Biceps was used heavily to draw forward the upper arm, hence requiring the triceps to counteract the torque on the elbow joint. This behavior shows that all muscles, including bi-articular ones, were used to generate movements, which was only possible by utilizing the learned muscle Jacobian in conjunction with the optimization procedure. The results of the quadratic optimization depend to a large degree on the objective function used. For this experiment, the Euclidean norm of all muscle forces was utilized for the optimization. By choosing different objective functions, other criteria could be integrated into the optimization procedure, leading to different force distributions.

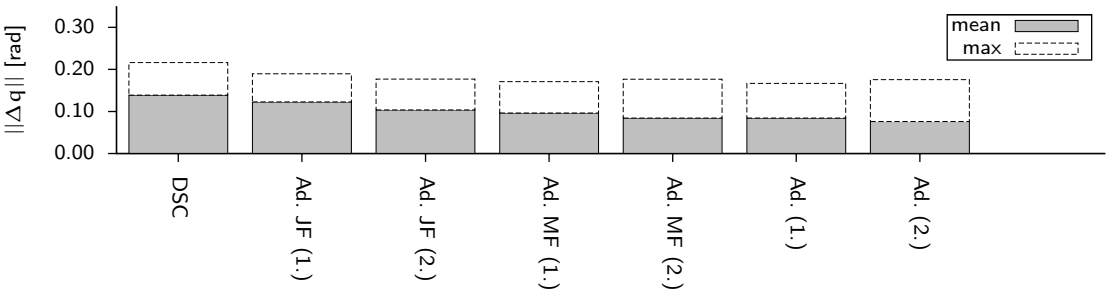


Fig. 8.9: Controller Comparison for full Anthrob. Mean and maximum absolute joint angle errors are shown for different controllers: (1) CFC, (2) DSC, and combinations (3-5) of the adaptive terms for joint friction (JF) and muscle friction (MF). Behavior of adaptive terms is depicted by showing errors of two consecutive executions.

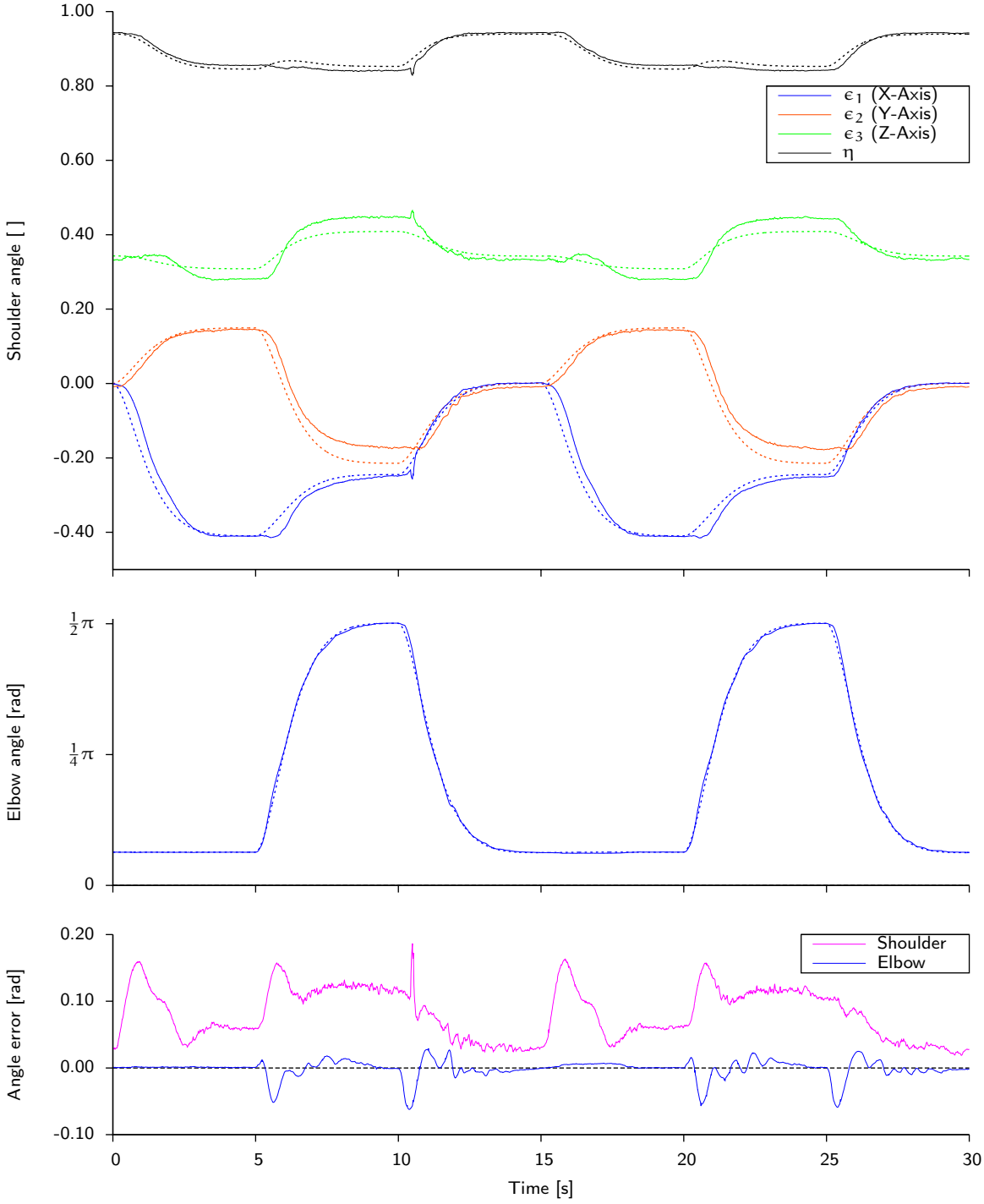


Fig. 8.10: Joint & Muscle Friction Compensation for full Anthrob. The reference trajectory (dashed) and the system output is shown for the quaternion of the spherical joint (top), followed by the simultaneously executed elbow trajectory (center) and the errors in both joints (bottom).

9 CONCLUSIONS

9.1 SUMMARY

Compared to existing tendon-driven robots, musculoskeletal robots feature several control challenges, due to which existing control techniques cannot be applied. In this work a control framework was developed to address these issues, starting from a generalized model for the class of musculoskeletal robots with electromagnetic actuators. This included modeling spherical joints in the context of standard dynamic algorithms to capture the dynamics of the stiff skeleton. These skeletal dynamics were extended by a dynamic model of the muscles, comprising the electromagnetic DC motor, the gearbox, the spindle, the tendon and the elastic element. Finally, the muscle kinematics which cover the geometric relationship between the muscle and the joint space were derived. In the presence of complex joints, multi-articular muscles and colliding muscle paths, analytic models for the muscle kinematics were extremely complex to obtain. Therefore, approximations based on Artificial Neural Networks (ANNs) were proposed, showing the necessary steps to obtain a reliable representation of the muscle Jacobian. Parameterization of this generic model therefore implies the specification of the skeletal kinematics and dynamics which can be extracted from CAD data, as well as the muscle parameters. The latter include the electrical and mechanical constants of the actuator and the gearbox, the spindle radius, as well as the stress-strain relationship of the elastic element. Finally, samples have to be gathered from the physical system to establish the muscle Jacobian by utilization of ANNs. Note that changes in the dynamic parameters of e. g. a muscle do not affect the geometric relationship of the muscle kinematics.

Previously, the method of feedback linearization in conjunction with independent low-level muscle control laws was utilized both in the field of tendon-driven robot control, as well as for biomechanical simulation models. The application of these techniques to musculoskeletal robots allowed for direct comparison against the novel control law based on Dynamic Surface Control (DSC), developed in this work. DSC, which is based on *backstepping*, allowed for systematically obtaining a cascade of control laws without neglecting the dynamics of the low-level controllers. This controller was further improved by utilizing adaptive neural network control for compensating friction, both in the joints, as well as in the muscle routing mechanism. Altogether, a framework for musculoskeletal robots was created, comprising several scalable control laws. The generic formulation of control laws and system dynamics allow for application of the proposed methods to the full class of musculoskeletal robots with electromagnetic actuation. A requirement is, however, the

availability of the necessary proprioceptive sensors. These include joint angle, muscle force and actuator position sensors.

Each of the developed controllers was evaluated on a musculoskeletal robot arm—the Anthrob—which was specifically developed for this purpose. Previous musculoskeletal robots showed impressive results for pure FFW control [46, 75, 88, 92, 107], however several problems inhibited the development of suitable feedback control laws. These included missing sensor modalities or unavailability of dynamic models due to hand manufactured skeletons. Therefore, a modular actuation unit, including the necessary sensing capabilities, was developed and integrated into the design. The anthropomorphic shapes were created by exploiting modern 3d printing techniques. The latter enabled the utilization of CAD models for obtaining the system dynamics. Furthermore, joint angle sensing was included for both joint types. This required an external stereo tracking system for the spherical joint which was developed by Wittmeier [139]. On the other hand, the Anthrob featured the necessary dynamic structures for evaluating musculoskeletal control techniques, namely a bi-articular muscle, a spherical joint, as well as muscles colliding with the skeleton.

Utilizing the Anthrob, it was shown that the conventional feedback linearization controllers perform well for simple systems like a single revolute joint with antagonistic actuation. However, when more complex structures like spherical joints are considered, the unmodeled dynamics of the low-level controllers lead to massively degraded trajectory tracking performance. This was greatly improved by the DSC controller. Best performance was achieved when both joint friction, as well as muscle friction compensators were utilized in the adaptive neural network controller. Therefore, it was possible to control the full Anthrob including the spherical shoulder, a revolute elbow joint and a bi-articular muscle with improved trajectory tracking performance. Especially the integration of the bi-articular Biceps produced interesting muscle tension distributions, where the involvement of the Biceps can be governed by the chosen objective function for the quadratic optimization.

9.2 FUTURE WORK

In the future, better trajectory tracking performance can be achieved by several possible improvements in the robot design. These include higher control frequencies through increased communication bandwidth, e. g. by utilizing the faster FlexRay bus instead of CAN. Furthermore, the limited bandwidth of the shoulder angle sensing should be increased to improve the rate of the joint angle measurements, as well as the quality of the numerical differentiation to obtain the angular velocities. This can be achieved, either by increasing the frame rate of the camera system, or by integrating other sensor modalities for directly measuring the pose of the spherical joint. Another possibility is to estimate the joint angles from the muscle lengths, similar to the receptors in the human sensory motor systems. For this purpose the previously learned mapping between the joint and

the muscle space can be utilized in conjunction with absolute actuator position measurements and accurate models of the elastic elements. When reliable measurements of all joint angles are available, the generic formulation of the models and the scalability of the controllers will allow for the application of the developed control schemes to more complex musculoskeletal setups.

It has been shown that there are correlations between neural activity in the motor cortex and movements in the operational space [41]. A formulation of an operational space control scheme within the developed framework would hence be beneficial to explain human motor control. For this purpose, the method of optimizing muscle forces for a given objective function should be extended to solve both for the muscle as well as the joint redundancies in a single optimization step. In such a setup, the effects of different objective functions on the quadratic optimization procedure can be investigated, comparing the resulting trajectories to movements of the human body. Similarly, the null-space in the muscle redundancies can be utilized for tasks orthogonal to the joint movements, like joint or operational space impedance or pretensioning of muscles. The latter can subsequently be exploited, e. g. in explosive movements like throwing or jumping.

Another point of interest for future research is the learning of the kinematic and dynamic model. Currently, learning is performed *offline* after sampling the system manually. We know that humans are able to learn the kinematics of the muscles, as well as a dynamic model of the body, by random muscle activities, i. e. motor babbling [110]. Subsequently these models are refined by performing goal-oriented reaching movements. Similarities between the Internal Dynamics Model Hypothesis (IDMH) and adaptive feedback error learning have been shown by comparison [133]. For adaptive friction compensation of both joint, as well as muscle friction, convergence could be shown in this work. These are however not the only possibilities for introducing adaptive terms into the DSC control law. Similar to the works of Kobayashi and Ozawa [63] or Le-Tien and Albu-Schäffer [70], the dynamic model for the skeleton could be added to the adaptation process. Investigating methods to subsequently learn a full model of the muscle kinematics, as well as the skeletal dynamics, based on the framework developed in this work, presents an interesting research task. Possibly this could lead a huge step towards the understanding of human motor control. Furthermore, an automatic online learning scheme of the muscle kinematics would also improve the applicability of the proposed control schemes to larger musculoskeletal robots, where manual sampling of the full workspace poses a very tedious task.

A

FOUNDATIONS & MATHEMATICAL BACKGROUND

A.1 POSITION AND ORIENTATION

The theory of how position and orientation are described in three dimensional space can be found in a variety of books, including the literature on robotics [19, 24, 119]. However, the notations vary which is why a brief overview of the theory is given in this section. Where possible, the notations of [119] are followed.

The transformation between a coordinate frame b and a can be described by a translational and a rotational component. The translational component is denoted by a 3×1 vector.

$$ {}^{a}\vec{p}_{b} = \begin{bmatrix} {}^{a}p_{b}^{x} \\ {}^{a}p_{b}^{y} \\ {}^{a}p_{b}^{z} \end{bmatrix} \tag{A.1} $$

Thus, a vector ${}^{b}\vec{x}$ in coordinate frame b is transformed into a by adding ${}^{a}\vec{p}_{b}$. The transformation between a rotated frame b into a are defined by multiplication with a 3×3 rotation matrix.

$$ {}^{a}\vec{x} = {}^{a}R_{b} \cdot {}^{b}\vec{x} \tag{A.2} $$

For rotations around a single axes the rotation matrices can be given as follows.

$$ R_{X}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{A.3} $$

$$ R_{Y}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{A.4} $$

$$ R_{Z}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.5} $$

The rotation between coordinate frame c and a is described by the rotation matrix ${}^{a}R_{c}$.

$$ {}^{a}R_{c} = {}^{a}R_{b} \cdot {}^{b}R_{c} \tag{A.6} $$

Arbitrary rotations can be expressed by combining rotations around multiple axes. Combining rotations about the three fixed axes of a reference frame, yields the so called fixed angle representation which can be parametrized by a minimum set of three angles. Another solution is to apply three consecutive rotations around relative axis, leading to another minimal representation, the so called Euler Angles. Commonly the rotations axis are given together with the form of representation (e. g. XYZ Euler Angles, ZYZ Euler angle, etc.).

The angle between two vectors ${}^{b}\vec{x}$ and ${}^{b}\vec{y}$ in coordinate frame b is not affected by the transformation to a different coordinate frame a.

$$ {}^{a}R_{b}\,{}^{b}\vec{x} \cdot {}^{a}R_{b}\,{}^{b}\vec{y} = {}^{b}\vec{x} \cdot {}^{b}\vec{y} \tag{A.7} $$

Therefore, it can be seen that rotation matrices are orthogonal, leading to the following property.

$$ ({}^{a}R_{b})^{T} \cdot {}^{a}R_{b} = I \qquad \Rightarrow \qquad ({}^{a}R_{b})^{-1} = ({}^{a}R_{b})^{T} = {}^{b}R_{a} \tag{A.8} $$

If coordinate frames are both translationally and rotationally displaced, it is helpful to be able to express this transformation with a single operation. This can be represented by a 4×4 transformation matrix

$$ {}^{a}T_{b} = \begin{bmatrix} {}^{a}R_{b} & {}^{a}\vec{p}_{b} \\ 0 & 1 \end{bmatrix} \tag{A.9} $$

while the transformation of ${}^{b}x$ is executed as follows.

$$ \begin{bmatrix} {}^{a}\vec{x} \\ 1 \end{bmatrix} = {}^{a}T_{b} \cdot \begin{bmatrix} {}^{b}\vec{x} \\ 1 \end{bmatrix} \tag{A.10} $$

In contrast to pure rotation matrices, transformation matrices are not orthogonal. This is due to the additional 1 in the last row. The inverse of a transformation can be given as follows.

$$ {}^{b}T_{a} = ({}^{a}T_{b})^{-1} = \begin{bmatrix} ({}^{a}R_{b})^{T} & {}^{b}\vec{p}_{a} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{b}R_{a} & -{}^{b}R_{a} \cdot {}^{a}\vec{p}_{b} \\ 0 & 1 \end{bmatrix} \tag{A.11} $$

Transformations between coordinate frames can be used to express the *pose*, i. e. position and orientation, of a rigid body in a reference frame. For this purpose a coordinate frame is fixed to the rigid body and the transformation describes this frame in the reference frame. Here, we are often not only interested in the position of the rigid body, but also in higher order derivatives, e. g. the velocity. Both the linear velocity $\dot{\vec{x}}$ as well as the angular velocity $\vec{\omega}$ is described as a 3×1 vector.

$$ \dot{\vec{x}} = [\dot{x}^{x} \quad \dot{x}^{y} \quad \dot{x}^{z}]^{T} \tag{A.12} $$

$$ \vec{\omega} = [\omega^{x} \quad \omega^{y} \quad \omega^{z}]^{T} \tag{A.13} $$

The linear velocity is simply the derivative of the positional displacement vector and can be expressed either in the moving coordinate frame of the rigid body or in the fixed reference coordinate frame. A relation between any Euler or fixed angle representation and the corresponding angular velocity can be derived by differentiating the three consecutive rotation matrices.

A.2 QUATERNION ALGEBRA

Rotations are generally expressed by a 3×3 matrix, while only three parameters are needed to describe the rotation around three axis. (see Section A.1). However, it can be shown that any of the minimal representations come with the problem of a singularity when the first and the last axis is aligned, the so called gimbal lock [119]. Furthermore, interpolation between two Euler or fixed angles is never smooth and natural. In computer simulations quaternions which are sometimes also called Euler parameters have been used early on for the latter reason [118]. Here, rotation is parameterized by four instead of three parameters. In control the quaternion representation poses several challenges. First, the positional component (represented by quaternions) has a higher dimensionality than its first and second derivatives, the velocity and the acceleration and therefore conversion between the quaternion derivative and the rotational velocity has to be frequently applied. Secondly, suitable error measures have to be defined, to capture the dynamics of quaternions and lead to convergence of the closed loop control systems. Despite these challenges, the preferred compact representation of orientation in 3 dimensions are quaternions. They have been mostly used in controlling the orientation of aerial or underwater vehicles [27] or of the end-effector of a robotic system [145]. A quaternion is defined as

$$Q = \eta + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k \tag{A.14}$$

while η is generally referred to as the real part and the vector $\vec{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^T$ describes the imaginary part. i, j and k are operators that satisfy the following rules

$$i^2 = j^2 = k^2 = ijk = -1, \tag{A.15}$$

$$ij = k, jk = i, ki = j, \tag{A.16}$$

$$ji = -k, kj = -i, ik = -j \tag{A.17}$$

while the conjugate of Q is defined as

$$\tilde{Q} = \eta - \epsilon_1 i - \epsilon_2 j - \epsilon_3 k \tag{A.18}$$

and its norm $\|Q\|$ is given as follows.

$$\|Q\| = \tilde{Q} \circ Q = \eta^2 - \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 = Q \circ \tilde{Q} \tag{A.19}$$

Note that the multiplication between quaternions is generally not commutative. It can be shown that the multiplication of two quaternions Q_1 and Q_2, can be expressed as a matrix multiplication

$$Q_1 \circ Q_2 = \begin{bmatrix} \eta_1 & -\vec{\epsilon}_1^T \\ \vec{\epsilon}_1 & \eta I_3 + S(\vec{\epsilon}_1) \end{bmatrix} \cdot \begin{bmatrix} \eta_2 \\ \vec{\epsilon}_2 \end{bmatrix} = \underline{Q}_1 \cdot \begin{bmatrix} \eta_2 \\ \vec{\epsilon}_2 \end{bmatrix} \tag{A.20}$$

$$Q_2 \circ Q_1 = \begin{bmatrix} \eta_1 & -\vec{\epsilon}_1^T \\ \vec{\epsilon}_1 & \eta I_3 - S(\vec{\epsilon}_1) \end{bmatrix} \cdot \begin{bmatrix} \eta_2 \\ \vec{\epsilon}_2 \end{bmatrix} = \underline{\tilde{Q}}_1 \cdot \begin{bmatrix} \eta_2 \\ \vec{\epsilon}_2 \end{bmatrix} \tag{A.21}$$

with respective matrices for the conjugate of Q_1 as $\underline{Q}_1^T$ and $\underline{\tilde{Q}}_1^T$ [115], while the operator $S(\cdot)$ denotes the skew-symmetric matrix that satisfies $S(\vec{x})\vec{v} = \vec{x} \times \vec{v}$.

$$S(\vec{x}) = \begin{bmatrix} 0 & -x_z & x_y \\ x_z & 0 & -x_x \\ -x_y & x_x & 0 \end{bmatrix} \tag{A.22}$$

Parameterization of rotations utilizes unit quaternions, of which the norm is confined to $\|Q\| = 1$. This can be visualized as a unit sphere in four dimensional space where unit quaternions are constraint to the surface of the sphere [118]. Any quaternion that is not on the sphere does not only rotate a vector but changes also its length. By expressing a vector ${}^b\vec{x}$ in quaternion notation it can be rotated by a unit quaternion Q, denoting the rotation ${}^aR_b$. For this purpose, the real part is taken as $\eta = 0$, leading to ${}^bX = 0 + {}^bx_x i + {}^bx_y j + {}^bx_z k$. The rotation is executed by pre- and postmultiplying Q and $\tilde{Q}$. A relationship for the rotation matrix ${}^aR_b$ is obtained by introducing Eq. A.20 and Eq. A.21.

$${}^aX = Q \circ {}^bX \circ \tilde{Q} \tag{A.23}$$

$$\begin{bmatrix} 0 \\ {}^a\vec{x} \end{bmatrix} = \underline{Q} \cdot \underline{\tilde{Q}}^T \cdot \begin{bmatrix} 0 \\ {}^b\vec{x} \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & {}^aR_b \end{bmatrix} \cdot \begin{bmatrix} 0 \\ {}^b\vec{x} \end{bmatrix} \tag{A.24}$$

The resulting rotation matrix can be written as follows.

$$R(Q) = \begin{bmatrix} 1-2(\epsilon_2^2+\epsilon_3^2) & 2(\epsilon_1\epsilon_2-\eta\epsilon_3) & 2(\epsilon_1\epsilon_3-\eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2-\eta\epsilon_3) & 1-2(\epsilon_1^2+\epsilon_3^2) & 2(\epsilon_2\epsilon_3-\eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3-\eta\epsilon_2) & 2(\epsilon_2\epsilon_3-\eta\epsilon_1) & 1-2(\epsilon_1^2+\epsilon_2^2) \end{bmatrix} \tag{A.25}$$

While obviously increasing the dimensionality of the description, the issue of singularities is eliminated. Furthermore, quaternions can be used for efficiently generating rotational movements at constant speed around a constant rotation axis (e. g. *SLERP* [118]).

Similar to any other parameterization of a rotation (see Section A.1), differentiating a quaternion does not directly yield the angular velocity vector ω. As a base for the angular velocity either the rotated x'y'z' coordinate frame or the fixed xyz coordinate frame can be

chosen, resulting in ω' or ω, respectively. The relationships for the quaternion derivative and the respective angular velocities can be derived, starting from the definition of the quaternion rotation [115],

$$^{a}X = Q \circ {}^{b}X \circ \tilde{Q} \tag{A.26}$$

$$^{b}X = \tilde{Q} \circ {}^{a}X \circ Q \tag{A.27}$$

whereas Eq. A.27 is introduced into the differentiation of Eq. A.26.

$$\begin{aligned} {}^{a}\dot{X} &= \dot{Q} \circ {}^{b}X \circ \tilde{Q} + Q \circ {}^{b}X \circ \dot{\tilde{Q}} \\ &= \dot{Q} \circ \tilde{Q} \circ {}^{a}X \circ Q \circ \tilde{Q} + Q \circ \tilde{Q} \circ {}^{a}X \circ Q \circ \dot{Q} \\ &= \dot{Q} \circ \tilde{Q} \circ {}^{a}X + \circ {}^{a}X \circ Q \circ \dot{Q} \end{aligned} \tag{A.28}$$

The scalar part of $\dot{Q} \circ \tilde{Q}$ and $Q \circ \dot{\tilde{Q}}$ is both zero, since Q is a unit quaternion and the vector parts are opposite.

$$\dot{Q} \circ \tilde{Q} = \begin{bmatrix} 0 \\ \vec{w} \end{bmatrix} \tag{A.29}$$

$$Q \circ \dot{\tilde{Q}} = \begin{bmatrix} 0 \\ -\vec{w} \end{bmatrix} \tag{A.30}$$

Utilizing this relationship, the quaternion differentiation can be simplified as follows.

$${}^{a}\dot{X} = \begin{bmatrix} 0 \\ \vec{w} \end{bmatrix} \circ {}^{a}X + {}^{a}X \circ \begin{bmatrix} 0 \\ -\vec{w} \end{bmatrix} = \begin{bmatrix} -\vec{w}^{T} \cdot {}^{a}\vec{x} \\ \vec{w} \times {}^{a}\vec{x} \end{bmatrix} + \begin{bmatrix} {}^{a}\vec{x}^{T} \cdot \vec{w} \\ \vec{w} \times {}^{a}\vec{x} \end{bmatrix} \tag{A.31}$$

$$= \begin{bmatrix} 0 \\ 2\vec{w} \times {}^{a}\vec{x} \end{bmatrix} \tag{A.32}$$

$${}^{a}\dot{x} = 2\vec{w} \times {}^{a}\vec{x} = \vec{\omega} \times {}^{a}\vec{x} \tag{A.33}$$

Hence, $\vec{\omega} = 2\vec{w}$. The four relationships for mapping quaternion derivatives to rotational velocities can be stated, utilizing Eq. A.20 and Eq. A.21.

$$\begin{bmatrix} 0 \\ \vec{\omega} \end{bmatrix} = 2 \cdot \underline{\tilde{Q}}^{T} \cdot \dot{Q} \quad = 2 \begin{bmatrix} \eta & \vec{\epsilon}^{T} \\ -\vec{\epsilon} & \eta \cdot I_3 + S(\vec{\epsilon}) \end{bmatrix} \cdot \dot{Q} \tag{A.34}$$

$$\dot{Q} = \frac{1}{2} \cdot \underline{\tilde{Q}} \cdot \begin{bmatrix} 0 \\ \vec{\omega} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \eta & -\vec{\epsilon}^{T} \\ \vec{\epsilon} & \eta \cdot I_3 - S(\vec{\epsilon}) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \vec{\omega} \end{bmatrix} \tag{A.35}$$

$$\begin{bmatrix} 0 \\ \vec{\omega'} \end{bmatrix} = 2 \cdot \underline{Q}^{T} \dot{Q} \quad = 2 \begin{bmatrix} \eta & \vec{\epsilon}^{T} \\ -\vec{\epsilon} & \eta \cdot I_3 - S(\vec{\epsilon}) \end{bmatrix} \cdot \dot{Q} \tag{A.36}$$

$$\dot{Q} = \frac{1}{2} \cdot \underline{Q} \cdot \begin{bmatrix} 0 \\ \vec{\omega'} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \eta & -\vec{\epsilon}^{T} \\ \vec{\epsilon} & \eta \cdot I_3 + S(\vec{\epsilon}) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \vec{\omega'} \end{bmatrix} \tag{A.37}$$

We can define $G(Q)$ and $U(Q)$ as follows

$$G(Q) = \begin{bmatrix} -\epsilon_1 & \eta & -\epsilon_3 & \epsilon_2 \\ -\epsilon_2 & \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_3 & -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \tag{A.38}$$

$$U(Q) = \begin{bmatrix} -\epsilon_1 & \eta & \epsilon_3 & -\epsilon_2 \\ -\epsilon_2 & -\epsilon_3 & \eta & \epsilon_1 \\ -\epsilon_3 & \epsilon_2 & -\epsilon_1 & \eta \end{bmatrix} \tag{A.39}$$

and state the relationships between the derivative of a quaternion $\dot{Q}$ and the angular velocities ω and ω', respectively.

$$\omega = 2 \cdot G(Q) \cdot \dot{Q} \tag{A.40}$$

$$\dot{Q} = \frac{1}{2} \cdot G(Q)^T \cdot \omega \tag{A.41}$$

$$\omega' = 2 \cdot U(Q) \cdot \dot{Q} \tag{A.42}$$

$$\dot{Q} = \frac{1}{2} \cdot U(Q)^T \cdot \omega' \tag{A.43}$$

By differentiating a unit quaternion, a four dimensional vector is generated that is tangential to the sphere of unit quaternions. Therefore, an integration step leads to a quaternion that is not on the sphere and is therefore not a valid rotation. This problem can be solved by normalization after each integration step.

$$Q_{k+1} = \frac{Q_k + \frac{dQ}{dt}\Delta t}{\|Q_k + \frac{dQ}{dt}\Delta t\|} \tag{A.44}$$

In theory, normalization is of no concern when quaternion multiplication is performed, as it can be easily proven that the multiplication of two unit quaternions leads again to a unit quaternion. However, due to numerical instabilities especially around the zero rotation, it might become necessary to normalize the result of a multiplication.

A.3 SPATIAL VECTOR ALGEBRA

There are several different notations for robot dynamics which include three dimensional vectors or 4×4 transformation matrices. Another elegant solution is the utilization of six dimensional vectors that comprise both the angular, as well as the translational components. These so called spatial vectors are subdivided in two groups, (1) the force related vectors, e. g. force or momentum, and (2) motion vectors comprising velocity and acceleration. This kind of description allows for the compact specification of dynamic algorithms. In this section only a basic overview of the topic is presented, to state the definitions

necessary for the subsequent descriptions of dynamic algorithms. For a broader definition, the reader is referred to the literature [24, 119]. In the following, spatial vectors are denoted, utilizing underlined variable names, to avoid confusion. The dot product for spatial vectors is only defined between motion and force vectors.

$$\underline{v} \cdot \underline{f} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \cdot \begin{bmatrix} \vec{\tau} \\ \vec{f} \end{bmatrix} = \vec{\omega} \cdot \vec{\tau} + \vec{v} \cdot \vec{f} \tag{A.45}$$

Similar to the definition of 4×4 transformation matrices (see Section A.1), spatial vector descriptions include a coordinate transform which is defined both for any motion vector $\underline{v}$ and any force vector $\underline{f}$ between a coordinate frame a and b,

$$^{b}\underline{v} = {}^{b}\underline{X}_{a}\, {}^{a}\underline{v} \tag{A.46}$$

$$^{b}\underline{f} = {}^{b}\underline{X}_{a}^{-T}\, {}^{a}\underline{f} \tag{A.47}$$

where ${}^{b}\underline{X}_{a}$ is a 6×6 dimensional matrix, which can be expressed using the rotation ${}^{b}R_{a}$, as well as the translation ${}^{b}p_{a}$.

$$^{b}\underline{X}_{a} = \begin{bmatrix} {}^{b}R_{a} & 0 \\ S({}^{b}p_{a})\,{}^{b}R_{a} & {}^{b}R_{a} \end{bmatrix} \tag{A.48}$$

The operator $S(\cdot)$, denotes the skew-symmetric matrix (see Eq. A.22). The inverse of a spatial transform is defined as follows.

$$^{b}\underline{X}_{a}^{-1} = {}^{a}\underline{X}_{b} = \begin{bmatrix} {}^{a}R_{b} & 0 \\ -{}^{a}R_{b}S({}^{b}p_{a}) & {}^{a}R_{b} \end{bmatrix} \tag{A.49}$$

Next to the dot product, there are two cross products defined, (1) between two motion vectors and (2) between a motion vector and a force vector.

$$\underline{v}_1 \times \underline{v}_2 = \begin{bmatrix} \omega_1 \times \omega_2 \\ v_1 \times \omega_2 + \omega_1 \times v_2 \end{bmatrix} \tag{A.50}$$

$$\underline{v} \times \underline{f} = \begin{bmatrix} \omega \times \tau + v \times f \\ \omega \times f \end{bmatrix} \tag{A.51}$$

The missing link between the motion and the force in dynamics is the inertia which in the spatial formulation comprises the mass m as well as the inertia J of a rigid body and is defined with respect to a specific coordinate frame a

$$^{a}\underline{I} = \begin{bmatrix} {}^{b}J + mS({}^{a}p_{b})S^{T}({}^{a}p_{b}) & mS({}^{a}p_{b}) \\ mS^{T}({}^{a}p_{b}) & mI_3 \end{bmatrix} \tag{A.52}$$

with a coordinate frame b located at the center of mass of the rigid body, in which the inertia ${}^{b}J$ is given. I_l denotes the $l \times l$ identity matrix. Hence, we can utilize this definition and state the equation of motion for a rigid body,

$$\underline{f} = J\underline{a} + \underline{v} \times J\underline{v} \tag{A.53}$$

whereas $\underline{a}$ denotes the spatial acceleration which is defined along the lines of the spatial velocity.

A.4 RECURSIVE NEWTON-EULER ALGORITHM

There exist several ways to obtain a dynamic model for robotic manipulators. One way is to express the equation of motion of the robot (see also Eq. 4.12) in a closed form, where a full analysis of the dynamic system, e. g. by the method of Lagrange [79], finally yields the model. However, it is also possible to capture the system in a more natural description and obtain the forward or inverse dynamics online for certain system states $q, \dot{q}$. The latter is not necessarily computationally more expensive, as there are algorithms to solve this problem in $\mathcal{O}(n)$. The recursive Newton-Euler algorithm (see Listing A.1) computes the inverse dynamics for a given robotic structure, utilizing spatial vector algebra for a more compact description of the dynamics of motion (see Section A.3). In the given form, the algorithm consists of two loops, one forward loop that is executed for each joint from 1 to the number of joints N (lines 7-25) and a second for the inverse loop traversing the kinematic chain from the end-effector to the base. The algorithm can be subdivided into four parts which can also be executed in separate loops if the state of the model is stored appropriately. First, the model is updated with the current state, given by the pose q and the joint velocity $\dot{q}$. Based on the type of the joint, a spatial transform for the joint $\underline{X}_J(i)$ is computed and used to compute the spatial transform from the previous joint to the current joint ${}^{i}\underline{X}_{p(i)}$ utilizing the transform for the interconnecting link $\underline{X}_L(i)$ (lines 10-11). In case the current joint is not the one closest to the base, the transform from the current joint to the base ${}^{i}\underline{X}_0$ is updated (line 13). This step is only necessary when external forces ${}^{0}\underline{f}_i^e$ are considered. Finally, the $6 \times n_i$ matrix Φ_i is updated, denoting the free modes of the current joint, for a joint with n_i DoF (line 15). It relates the $n_i \times 1$ vector of the current joint velocity $\dot{q}_i$ with the spatial velocity of the joint movement, given in the joint frame.

$$\underline{v}_J(i) = \Phi_i \dot{q} \tag{A.54}$$

Subsequently, the partial derivative $\mathring{\Phi}_{C_i}$ (line 16) is updated which is defined as follows.

$$\mathring{\Phi}_i = \frac{\partial \Phi_i}{\partial q_i} \dot{q}_i \tag{A.55}$$

However, for most known joint types, Φ_i is a constant matrix and therefore $\mathring{\Phi}_i$ evaluates to zero. Therefore, it is possible to omit this step for most robots.

Listing A.1: Recursive Newton-Euler Algorithm in link coordinates for inverse dynamics [119]. The algorithm can be subdivided into four parts, (1) setting the position dependent variables, (2) traversing the kinematic structure in forward direction and passing the velocity, (3) traversing the kinematic structure in forward dircection and passing the acceleration, and finally (4) traversing the kinematic structure in inverse direction and passing the forces. On the last pass, the joint torques τ_i are calculated from the spatial forces, acting on the joints.

```
inputs: q, q̇, q̈, model, ⁰f_i^e
outputs: τ
model data: N, jtype(i), p(i), X_L(i), J_i

v_0 = 0
a_0 = −a_g
for i = 1 to N do

  // setting position dependent variables
  X_J(i) = xjcalc(jtype(i),q_i))
  ⁱX_p(i) = X_J(i) X_L(i)
    if p(i) ≠ 0 then
  ⁱX_0 = ⁱX_p(i) ᵖ⁽ⁱ⁾X_0
  end
  Φ_i = pcalc(jtype(i),q_i)
  Φ̊_Ci = pdcalc(jtype(i),q_i,q̇_i)

  // forward velocity
  v_i = ⁱX_p(i) v_p(i) + Φ_i q̇_i          // pass spatial velocity from previous joint

  // forward acceleration
  ζ_i = Φ̊_Ci q̇_i + v_i × Φ_i q̇_i
  a_i = ⁱX_p(i) a_p(i) + Φ_i q̈_i + ζ_i      // pass spatial acceleration from previous joint
  f_i = J_i a_i + v_i × J_i v_i − ⁱX_0^−T ⁰f_i^e  // calculate the spatial force at the joint
end

// inverse force
for i = N to 1 do
  τ_i = Φ_i^T f_i                        // compute joint torque from spatial force
  if p(i) ≠ 0 then
    f_p(i) = f_p(i) + ⁱX_p(i)^T f_i       // pass spatial force to previous joint
  end
end
```

The second pass is responsible for updating the velocities which is done by traversing the robot and subsequently applying the joint velocity $\dot{q}_i$ (line 19). Third, the same is done for the acceleration (line 22-23) and the spatial forces necessary for these accelerations are computed (line 24). In the last pass which traverses the kinematic chain from the end-effector to the base, joint torques are computed from the spatial forces (line 29). These spatial forces are handed through from one joint to the next until the base is reached (line 31).

Thus, the recursive Newton-Euler algorithm provides an efficient method of computing the inverse dynamics. The model can be stated in an intuitive description of joint types and interconnecting links, comprising the kinematics, i. e. the transform from one joint to the next (see Fig. 4.1), as well as the dynamic properties like mass and inertia. This description can be used by the algorithm to compute joint torques for a given trajectory.

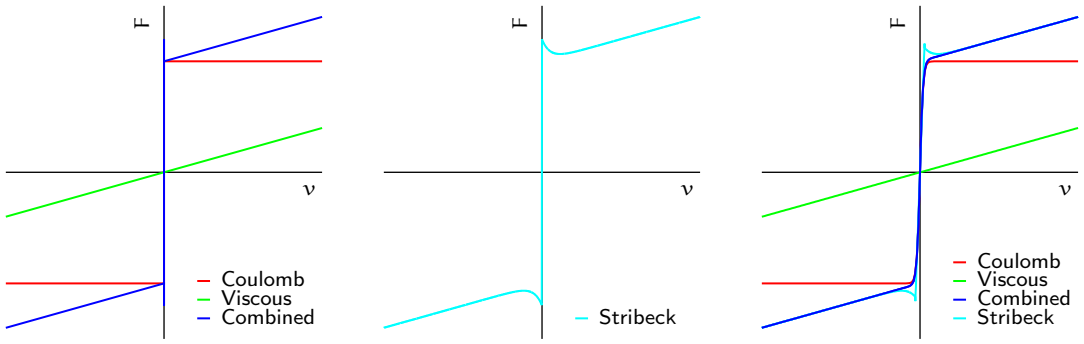
However, in some cases, the inertia matrix, Coriolis and centrifugal components and gravitational terms are need explicitly. If this is the case, several passes through the recursive Newton-Euler algorithm are required. The mass matrix $H(q)$ can be computed by first setting the joint velocities and the gravity vector to zero to eliminate Coriolis, centrifugal and gravity effects. Subsequently, each column in the mass matrix is computed by setting the corresponding joint acceleration to one. Therefore, one evaluation of the recursive Newton-Euler algorithm is required for each joint. The vector of Coriolis and centrifugal component $C(q, \dot{q})\dot{q}$, can be obtained by setting gravity and acceleration to zero, hence eliminating the gravity terms and the mass matrix. Similarly, the vector of gravity terms is computed by setting joint velocities and accelerations to zero. If necessary, each term of the equation of motion can thus be computed individually for a certain pose, however, as multiple passes of the recursive Newton-Euler algorithm are required, the complexity is increased for this approach. Since the complexity of the basic approach is $\mathcal{O}(n)$ and it needs to be executed n times for the mass matrix $H(q)$ and once for each $C(q, \dot{q})\dot{q}$ and $\tau_G(q)$, the overall complexity evaluates to $\mathcal{O}(n^2 + n + n) = \mathcal{O}(n^2)$.

A.5 FRICTION MODELS

The field of tribology which deals with modeling the effects of friction between two surfaces, has produced a wide range of models on how the friction force is affected by lubrication, the normal force f_n acting to press the two surfaces together, the relative velocity between surfaces, etc. When dynamic models of robots are developed, characterizing these effects is crucial. In the following, models for viscous and static friction as well as stiction are described.

Viscous friction describes the effect where the friction force depends only on the velocity v between the two surfaces. This model is used to describe the effects of friction when the surfaces are well lubricated and friction is mainly caused by hydrodynamic effects in the lubricator with a constant viscous friction coefficient μ_v.

$$f_v = \mu_v \cdot v \tag{A.56}$$



(a) Classic Model (b) Stribeck (c) Approximation

Fig. A.1: Friction Models. The force over velocity relationship is shown in comparison, for (a) the classic model, consisting of Coulomb and viscous and stiction, (b) the more advanced model of Stribeck and (c) an approximation that is numerically more stable for both the classic and the Stribeck model.

The simplest model for dry friction is described by the Coulomb friction. Here the friction force depends only on the normal force f_n, and a constant Coulomb friction coefficient μ_c.

$$f_c = \mu_c \cdot f_n \cdot \operatorname{sgn}(v) \tag{A.57}$$

When both surfaces are at rest, the effect of stiction is observed. Then the friction force is counter-acting the sum of all other forces $\sum f$ applied until these forces overcome stiction and the surfaces break free [96].

$$f_s = \begin{cases} -\sum f & \text{if} \quad \sum f < \mu_s \cdot f_n \\ -\mu_s \cdot f_n \cdot \operatorname{sgn}(\sum f) & \text{if} \quad \sum f \geqslant \mu_s \cdot f_n \end{cases} \tag{A.58}$$

The three models of friction can be combined to form the classic model of friction (see Fig. A.1a), however in numerical representations it needs to be decided when the velocity is zero. In simulations this is usually solved by the Karnopp model [57] where a zero velocity interval $|v| < \epsilon$ is used to make that decision. However, even though this leads to good simulation results in most cases, this does not agree with real friction. A more realistic behavior is obtained by the Stribeck model (see Fig. A.1b) [124].

$$f_F = f_c + (f_s - f_c) \cdot e^{-|v/v_s|^{\delta_s}} + \mu_v \cdot v \tag{A.59}$$

It is observed that neither Eq. A.59 nor the combined models of standard friction are continuous at $v = 0$. In simulation or in control where a model might be used to compensate for the effects of friction, this can quickly lead to instabilities. Therefore, an approximation around $v = 0$ can be utilized both for the Coulomb as well as the Stribeck friction

model. In the first, the sigmoid function is utilized, while the parameter a can be used to scale the transition through zero (see Fig. A.1c).

$$f_F = \mu_c \cdot f_n \cdot \left(\frac{2}{1 + e^{-a \cdot v}} - 1 \right) \tag{A.60}$$

Note that stiction is neglected here. To address this, another approximation leads to the extended Stribeck model which is obtained by adding a linear function for the transition through zero (see Fig. A.1c), which is evaluated within a small range $\pm\epsilon$, similar to the Karnopp model.

$$f_F = \begin{cases} \frac{f_s}{\epsilon} \cdot v & \text{if} \quad |v| < \epsilon \\ f_c + (f_s - f_c) \cdot e^{-|v/v_s|^{\delta_s}} + f_v \cdot v & \text{otherwise} \end{cases} \tag{A.61}$$

Friction models of the given types can be added to e. g. the equation of motion (see Eq. 4.12) by formulating a friction torque $\tau_F(q, \dot{q}, \tau)$.

B

SUPPLEMENTARY DATA

Table B.1: Anthrob Actuators. Properties of the brushed DC motor and gearbox combinations by Maxon Motors are shown.

		τ_{nom} [mNm]	speed [rpm]	gearb. ratio	gearb. eff.
Elbow	A-max22 + GP22HP	6.8	5240	157:1	59 %
Shoulder & Biceps	RE-25 + GP22HP	26.2	8240	128:1	59 %
Hand	RE-16 + GP16A	2.37	8230	370:1	65 %

Table B.2: Anthrob Muscle Parameters.

Name	Joints	NBR rings		
		l [mm]	d [mm]	quantity
Anterior Deltoid	Shoulder	16.00	5.00	2
Lateral Deltoid	Shoulder	15.00	5.00	2
Posterior Deltoid	Shoulder	15.00	4.00	3
Pectoralis	Shoulder	10.00	5.00	1
Teres Major	Shoulder	12.00	4.00	3
Teres Minor	Shoulder	10.69	3.80	3
Supra	Shoulder	10.69	3.53	3
Infraspinatus	Shoulder	10.69	3.80	3
Biceps	Shoulder & Elbow	23.00	4.00	3
Brachialis	Elbow	20.00	3.00	3
Triceps	Elbow	13.00	4.00	1

where l is the inner diameter and d the cross section diameter of the NBR rings.

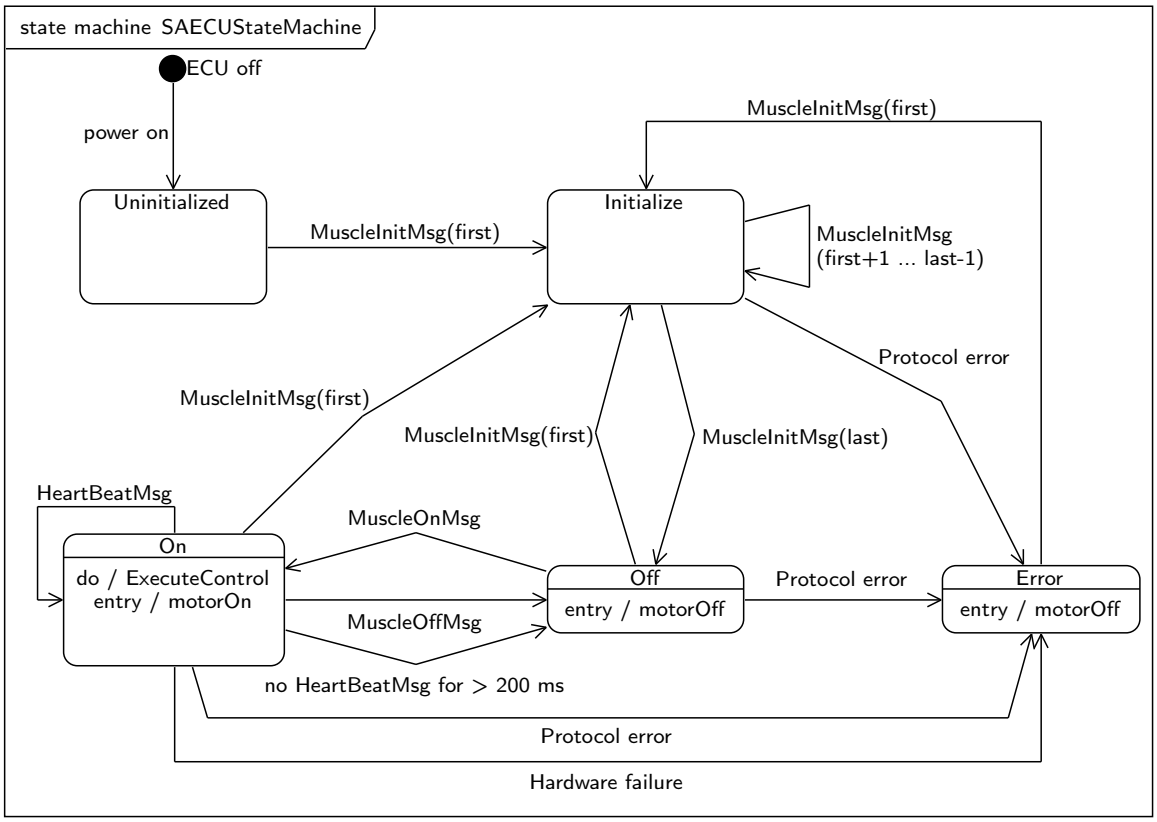


Fig. B.1: ECU State Machine. A diagram of the state machine that is implemented on the distributed ECUs for each of the muscles to form a fail-silent unit is shown.

Table B.3: Simulation Model Parameters.

Skeletal Dynamics:	
mass [kg]	1.00
length [m]	0.20
diameter [m]	0.02
J_{xx} [kg m^2]	$3.37 \cdot 10^{-3}$
J_{yy} [kg m^2]	$3.37 \cdot 10^{-3}$
J_{zz} [kg m^2]	$6.67 \cdot 10^{-5}$

Joint Friction:	low	high
τ_v [Nm/s]	0.02	0.02
ϵ [ ]	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$
τ_s [Nm]	0.01	0.05
τ_c [Nm]	0.008	0.04
ω_s [rad/s]	0.05	0.05

Muscle Dynamics:	
R_A [Ω]	0.516
L_A [H]	$3.08 \cdot 10^{-5}$
c_m [Nm/A]	$11.5 \cdot 10^{-3}$
J_M [kg m^2]	$14.3 \cdot 10^{-7}$
r_G [ ]	128
c_F [ ]	0.59
μ_v [Nm/rad s]	$1.5 \cdot 10^{-5}$
J_G [kg m^2]	$0.4 \cdot 10^{-7}$
R_S [m]	0.006
K [N/m]	5000

Muscle Friction:	low	high
μ_c [ ]	0.0	0.4

Muscle Kinematics:	1	2	3	4	5
1st Muscle Anchor[1] ${}^{a}x_1$ [m]	$\begin{bmatrix} 0.03 \\ 0.01 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.03 \\ -0.01 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.03 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0.03 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.03 \\ 0 \end{bmatrix}$
2nd Muscle Anchor[1] ${}^{b}x_2$ [m]	$\begin{bmatrix} 0 \\ 0.01 \\ 0.03 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.01 \\ 0.03 \end{bmatrix}$	$\begin{bmatrix} -0.01 \\ 0 \\ 0.03 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0.01 \\ 0.03 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.01 \\ 0.03 \end{bmatrix}$

[1] Muscle anchor points are defined in frames a and b, which are located in the spherical shoulder joint, according to Fig. 6.2b. While a is fixed in the base, b moves with the movable link.

Table B.4: Simulative Control Parameters.

Trajectory Computation:		Adaptive Joint Friction:	
P	$I_3 \cdot 5$	Γ_{JF}	-0.05
D	$I_3 \cdot 4$	Σ_{JF}	0.1
		τ_{Fmin} [Nm]	-0.06
Control Period:		τ_{Fmax} [Nm]	0.06
High-level [s]	$5 \cdot 10^{-3}$		
Low-Level [s]	$1 \cdot 10^{-3}$		
Backstepping:		Adaptive Muscle Friction:	
K_d	$I_3 \cdot 2$	Γ_{MF}	80
Λ_1	$I_3 \cdot 15$	Σ_{MF}	0.1
Λ_2	$I_5 \cdot 15$	f_{Fmin} [N]	0
Λ_3	$I_5 \cdot 150$	f_{Fmax} [N]	15
DSC:			
μ_1 [s]	0.01		
μ_2 [s]	0.005		

Table B.5: Comparison of Simulation Results.

	force sensor	friction	1^{st} run				2^{nd} run			
			mean		max		mean		max	
			$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]
Backstepping	de	low	0.006	0.005	0.120	0.022	0.006	0.005	0.102	0.022
DSC			0.005	0.004	0.120	0.015	0.004	0.004	0.093	0.015
DSC	le		0.008	0.014	0.273	0.030	0.009	0.015	0.274	0.030
DSC	de	high	0.021	0.069	0.335	0.130	0.022	0.069	0.335	0.130
Ad. JF			0.015	0.035	0.224	0.088	0.011	0.025	0.204	0.048
Ad. MF			0.015	0.028	0.245	0.061	0.020	0.035	0.233	0.073
Ad. (both)			0.008	0.011	0.172	0.037	0.007	0.009	0.172	0.026

Table B.6: Anthrob Shoulder Control Parameters.

Trajectory Computation:		CFC:	
P	$I_3 \cdot 2$	P	600
D	$I_3 \cdot 2.5$	D	30
Control Period:		Adaptive Joint Friction:	
High-level [s]	$5 \cdot 10^{-3}$	Γ_{JF}	-0.005
Low-Level [s]	$1 \cdot 10^{-3}$	Σ_{JF}	0.1
		τ_{Fmin} [Nm]	-0.04
Backstepping:		τ_{Fmax} [Nm]	0.04
K_d	$I_3 \cdot 15$		
Λ_1	$I_3 \cdot 10$	Adaptive Muscle Friction:	
Λ_2	$I_8 \cdot 15$	Γ_{MF}	60
Λ_3	$I_8 \cdot 170$	Σ_{MF}	0.1
		f_{Fmin} [N]	0
DSC:		f_{Fmax} [N]	40
μ_1 [s]	0.02		
μ_2 [s]	0.01	High-gain observer:	
		h_1	120
		h_2	120

Table B.7: Comparison of Anthrob Shoulder Results.

	1^{st} run				2^{nd} run			
	mean		max		mean		max	
	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]
CFC	0.259	0.241	2.628	0.428	0.263	0.271	2.502	0.476
DSC	0.224	0.157	2.530	0.222	0.185	0.129	2.164	0.218
Ad. JF	0.211	0.148	2.168	0.213	0.187	0.117	2.382	0.198
Ad. MF	0.214	0.106	2.615	0.179	0.193	0.104	2.274	0.193
Ad. (both)	0.228	0.097	2.497	0.175	0.178	0.077	2.624	0.172

Listing B.1: Anthrob Model. A description of the model is given in XML format, as used by the Robotics Library (RL) by Markus Rickert. The kinematic relation between the joints (<revolute> and <spherical>) is described by defining frames and the <fixed> transformations in between. A frame can either be defined as a <frame> or a <body> with dynamic properties, i. e. a center of mass, an inertia and a mass. [109]

```
<?xml version="1.0" encoding="UTF-8"?>
<rlmdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="rlmdl.xsd">
        <model>
                <manufacturer>Awtec</manufacturer>
                <name>AnthrobNoBiceps</name>
                <world id="frame0">
                        <rotation>
                                <x>0</x>
                                <y>0</y>
                                <z>0</z>
                        </rotation>
                        <translation>
                                <x>0</x>
                                <y>0</y>
                                <z>0</z>
                        </translation>
                        <g>
                                <x>0</x>
                                <y>0</y>
                                <z>9.81</z>
                        </g>
                </world>
                <frame id="frameTurningPoint" />
                <body id="upperArm">
                        <cm>
                                <x>-0.0003</x>
                                <y>0.0025</y>
                                <z>-0.1257</z>
                        </cm>
                        <i>
                                <xx>2.53e-03</xx>
                                <yy>2.50e-03</yy>
                                <zz>6.45e-04</zz>
                                <yz>2.56e-05</yz>
                                <xz>1.23e-06</xz>
                                <xy>2.45e-04</xy>
                        </i>
                        <m>0.704</m>
                </body>
                <frame id="frameElbowAxis" />
```

```
<frame id="foreArmPre" />
<body id="foreArm">
        <cm>
                <x>0.0016</x>
                <y>0.0012</y>
                <z>-0.081</z>
        </cm>
        <i>
                <xx>4.04e-04</xx>
                <yy>4.0e-04</yy>
                <zz>1.47e-05</zz>
                <yz>-2.82e-06</yz>
                <xz>-1.01e-05</xz>
                <xy>2.30e-07</xy>
        </i>
        <m>0.111</m>
</body>
<body id="endeffectorLoad">
        <cm>
                <x>0</x>
                <y>0</y>
                <z>0</z>
        </cm>
        <i>
                <xx>0</xx>
                <yy>0</yy>
                <zz>0</zz>
                <yz>0</yz>
                <xz>0</xz>
                <xy>0</xy>
        </i>
        <m>0</m>
</body>
<fixed id="linkBase">
        <frame>
                <a idref="frameo" />
                <b idref="frameTurningPoint" />
        </frame>
        <rotation>
                <x>0</x>
                <y>0</y>
                <z>0</z>
        </rotation>
        <translation>
                <x>0</x>
                <y>-0.218</y>
                <z>0.410</z>
```

```
                </translation>
        </fixed>
        <spherical id="shoulder">
                <frame>
                        <a idref="frameTurningPoint" />
                        <b idref="upperArm" />
                </frame>
        </spherical>
        <fixed id="linkUpper">
                <frame>
                        <a idref="upperArm" />
                        <b idref="frameElbowAxis" />
                </frame>
                <rotation>
                        <x>90</x>
                        <y>0</y>
                        <z>0</z>
                </rotation>
                <translation>
                        <x>0</x>
                        <y>0</y>
                        <z>-0.217</z>
                </translation>
        </fixed>
        <revolute id="elbow">
                <frame>
                        <a idref="frameElbowAxis" />
                        <b idref="foreArmPre" />
                </frame>
        </revolute>
        <fixed id="ElbowPre">
                <frame>
                        <a idref="foreArmPre" />
                        <b idref="foreArm" />
                </frame>
                <rotation>
                        <x>-90</x>
                        <y>0</y>
                        <z>0</z>
                </rotation>
                <translation>
                        <x>0</x>
                        <y>0</y>
                        <z>0</z>
                </translation>
        </fixed>
        <fixed id="linkForeArm">
```

```
                        <frame>
                                <a idref="foreArm" />
                                <b idref="endeffectorLoad" />
                        </frame>
                        <rotation>
                                <x>0</x>
                                <y>0</y>
                                <z>0</z>
                        </rotation>
                        <translation>
                                <x>0</x>
                                <y>0</y>
                                <z>-0.237</z>
                        </translation>
                </fixed>
        </model>
</rlmdl>
```

Table B.8: Anthrob Control Parameters.

Trajectory Computation:		Adaptive Joint Friction:	
P	$I_4 \cdot 2$	Γ_{JF}	-0.005
D	$I_4 \cdot 2.5$	Σ_{JF}	0.1
		τ_{Fmin} [Nm]	-0.04
Control Period:		τ_{Fmax} [Nm]	0.04
High-level [s]	$5 \cdot 10^{-3}$		
Low-Level [s]	$1 \cdot 10^{-3}$	Adaptive Muscle Friction:	
		Γ_{MF}	60
Backstepping:		$^{\circ}_{MF}$	0.1
K_d	$I_4 \cdot 15$	f_{Fmin} [N]	0
Λ_1	$I_4 \cdot 10$	f_{Fmax} [N]	$0.25 \cdot f_{max}$ [1]
Λ_2	$I_8 \cdot 15$		
Λ_3	$I_8 \cdot 170$	High-gain observer:	
		h_1	60
DSC:		h_2	60
μ_1 [s]	0.02		
μ_2 [s]	0.01		

[1] maximum forces [N]: $f_{max} = [50 \quad 50 \quad 50 \quad 50 \quad 50 \quad 90 \quad 90 \quad 80 \quad 170 \quad 80 \quad 50]^T$

Table B.9: Comparison of Anthrob Results.

	1^{st} run				2^{nd} run			
	mean		max		mean		max	
	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]	$\Delta\dot{q}$ [rad/s]	Δq [rad]
DSC	0.076	0.139	1.278	0.216	0.087	0.119	1.428	0.212
Ad. JF	0.090	0.123	1.530	0.190	0.079	0.104	1.205	0.177
Ad. MF	0.085	0.096	1.459	0.171	0.090	0.084	1.519	0.177
Ad. (both)	0.080	0.084	1.457	0.167	0.085	0.076	1.433	0.176

BIBLIOGRAPHY

[1] M. E. Abdallah, R. Platt Jr., C. W. Wampler, and B. Hargrave. Applied Joint-Space Torque and Stiffness Control of Tendon-Driven Fingers. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 74–79, 2010. ISBN 9781424486908. doi: 10.1109/ICHR.2010.5686289.

[2] M. E. Abdallah, R. Platt, and C. W. Wampler. Decoupled torque control of tendon-driven fingers with tension management. *The International Journal of Robotics Research*, 32(2):247–258, Feb. 2013. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364912468302.

[3] A. Albu-Schäffer. *Regelung von Robotern mit elastischen Gelenken am Beispiel der DLR-Leichtbauarm*. PhD thesis, Technische Universität München, 2001.

[4] A. Albu-Schäffer and G. Hirzinger. A globally stable state feedback controller for flexible joint robots. *Advanced Robotics*, 15:799–814, 2001.

[5] A. Albu-Schäffer, C. Ott, and G. Hirzinger. A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. *Internation Journal of Robotics Research*, 26(1):23–39, 2007. ISSN 0278-3649. doi: http://dx.doi.org/10.1177/0278364907073776.

[6] A. Albu-Schäffer, O. Eiberger, M. Grebenstein, S. Haddadin, C. Ott, T. Wimbock, S. Wolf, and G. Hirzinger. Soft robotics. *IEEE Robotics & Automation Magazine*, 15 (3):20–30, 2008. doi: 10.1109/MRA.2008.927979.

[7] A. Albu-Schäffer, S. Wolf, O. Eiberger, S. Haddadin, F. Petit, and M. Chalon. Dynamic modelling and control of variable stiffness actuators. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 2155–2162, 2010. doi: 10.1109/ROBOT.2010.5509850.

[8] E. Alpaydin. *Introduction to machine learning*. MIT press, 2 edition, 2004.

[9] S. Arimoto and M. Sekimoto. Human-like movements of robotic arms with redundant DOFs: virtual spring-damper hypothesis to tackle the Bernstein problem. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 1860–1866, May 2006. ISBN 0-7803-9505-0. doi: 10.1109/ROBOT.2006.1641977.

[10] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer, 2 edition, May 1989. ISBN 9780387968902.

[11] Y. Asano, H. Mizoguchi, T. Kozuki, Y. Motegi, M. Osada, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Lower thigh design of detailed musculoskeletal humanoid "Kenshiro". In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 4367–4372, Oct. 2012. ISBN 978-1-4673-1736-8. doi: 10.1109/IROS.2012.6386225.

[12] K. J. Aström and B. Wittenmark. *Adaptive control*. Dover Publications, 2 edition, 2008. ISBN 9780486462783.

[13] C. M. Bishop. *Pattern recognition and machine learning*. Springer New York, 4 edition, 2006.

[14] B. Brogliato, R. Ortega, and R. Lozano. Global tracking controllers for flexible-joint manipulators: a comparative study. *Automatica*, 31(7):941–956, July 1995. ISSN 0005-1098. doi: 10.1016/0005-1098(94)00172-F.

[15] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991. doi: DOI:10.1016/0004-3702(91)90053-M.

[16] D. G. Caldwell, G. A. Medrano-Cerda, and M. Goodwin. Control of pneumatic muscle actuators. *IEEE Control Systems*, 15(1):40–48, Feb. 1995. ISSN 1066-033X. doi: 10.1109/37.341863.

[17] Z. Chen, N. Y. Lii, T. Wimboeck, S. Fan, M. Jin, C. H. Borst, and H. Liu. Experimental Study on Impedance Control for the Five-Finger Dexterous Robot Hand DLR-HIT II. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 2010.

[18] E. Coumans. Bullet Physics Library. URL http://www.bulletphysics.com.

[19] J. J. Craig. *Introduction to Robotics*. Pearson, 3 edition, 2005.

[20] G. Dantzig. *Linear programming and extensions*. Princeton University Press, 1998.

[21] V. De Sapio, K. Holzbaur, and O. Khatib. The control of kinematically constrained shoulder complexes: physiological and humanoid examples. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 2952–2959, 2006. doi: 10.1109/ROBOT.2006.1642150.

[22] V. De Sapio, J. Warren, and O. Khatib. Predicting reaching postures using a kinematically constrained shoulder model. In *Advances in Robot Kinematics*, pages 209–218. Springer Netherlands, 2006.

[23] S. Fang, D. Franitza, M. Torlo, F. Bekes, and M. Hiller. Motion control of a tendon-based parallel manipulator using optimal tension distribution. *IEEE/ASME Transactions on Mechatronics*, 9(3):561–568, 2004. doi: 10.1109/TMECH.2004.835336.

[24] R. Featherstone. *Rigid Body Dynamics*. Springer Science+Business Media, LLC, 2008.

[25] A. G. Feld'man. On the functional tuning of the nervous system in movement control or preservation of stationary pose. II. Adjustable parameters in muscles. *Biofizika*, 11(3):498–508, 1966.

[26] Festo. Airic's arm, Jan. 2007. URL http://www.festo.com/cms/en_corp/9785.htm.

[27] O.-E. Fjellstad and T. I. Fossen. Quaternion feedback regulation of underwater vehicles. In *Proc. IEEE Conference on Control Applications*, pages 857 –862, Aug. 1994. doi: 10.1109/CCA.1994.381209.

[28] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. Prentice Hall, 1997.

[29] A. Gaschler. Visual motion capturing for kinematic model estimation of a humanoid robot. In *Proc. 33rd DAGM Symposium on Pattern Recognition*, pages 438–443. Springer, 2011.

[30] S. S. Ge and C. C. Hang. Direct adaptive neural network control of robots. *International Journal of Systems Science*, 27(6):533–542, 1996. ISSN 0020-7721. doi: 10.1080/00207729608929247.

[31] S. S. Ge and C. Wang. Direct adaptive NN control of a class of nonlinear systems. *IEEE Transactions on Neural Networks*, 13(1):214–221, 2002.

[32] S. S. Ge, T. H. Lee, and C. J. Harris. *Adaptive neural network control of robotic manipulators*, volume 19. World Scientific Publishing Company, 1998.

[33] S. S. Ge, T. H. Lee, and S. X. Ren. Adaptive friction compensation of servo mechanisms. *International Journal of Systems Science*, 32(4):523–532, Jan. 2001. ISSN 0020-7721. doi: 10.1080/00207720119378.

[34] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, 27(1):1–33, 1983.

[35] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics*. Addison-Wesley, 3 edition, 2001. ISBN 0201657023.

[36] H. Gray. *Anatomy of the Human Body*. Bartleby.com, 1918.

[37] M. Grebenstein, M. Chalon, W. Friedl, S. Haddadin, T. Wimböck, G. Hirzinger, and R. Siegwart. The hand of the DLR Hand Arm System: Designed for interaction. *The International Journal of Robotics Research*, 31(13):1531–1555, Nov. 2012. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364912459209.

[38] L. Gregoire, H. Veeger, P. Huijing, and G. van Ingen Schenau. Role of Mono- and Biarticular Muscles in Explosive Movements. *International Journal of Sports Medicine*, 05(06):301–305, Mar. 1984. ISSN 0172-4622, 1439-3964. doi: 10.1055/s-2008-1025921.

[39] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Safety evaluation of physical human-robot interaction via crash-testing. In *Proceedings of Robotics: Science and Systems RSS*, pages 217–224, 2007.

[40] G. Herrmann, S. S. Ge, and G. Guo. Practical implementation of a neural network controller in a hard disk drive. *IEEE Transactions on Control Systems Technology*, 13 (1):146–154, Jan. 2005. ISSN 1063-6536. doi: 10.1109/TCST.2004.838567(410)13.

[41] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, and J. P. Donoghue. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398): 372–375, May 2012. ISSN 0028-0836. doi: 10.1038/nature11076.

[42] N. Hogan. Impedance control : An Approach to Manipulation : Part I-Part III. *Journal of Dynamic Systems, Measurement, and Control*, 107(2):1–24, 1985.

[43] O. Holland and R. Knight. The Anthropomimetic Principle. In *Adaptation in Artificial and Biological Systems*, 2006.

[44] K. Holzbaur, W. M. Murray, and S. L. Delp. A Model of the Upper Extremity for Simulating Musculoskeletal Surgery and Analyzing Neuromuscular Control. *Annals of Biomedical Engineering*, 33(6):829–840, June 2005. ISSN 0090-6964. doi: 10.1007/s10439-005-3320-7.

[45] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[46] K. Hosoda, S. Sekimoto, Y. Nishigori, S. Takamuku, and S. Ikemoto. Anthropomorphic Muscular-Skeletal Robotic Upper Limb for Understanding Embodied Intelligence. *Advanced Robotics*, 26(7):729–744, 2012.

[47] Igus. Igus Robolink, 2011. URL http://www.igus.de/_Product_Files/Download/pdf/robolink_08_2012_mp_s.pdf.

[48] P. A. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice Hall, Upper Saddle River, NJ, 1996.

[49] S. C. Jacobsen, E. K. Iversen, D. Knutti, R. Johnson, and K. Biggers. Design of the Utah/M.I.T. Dextrous Hand. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 1520–1532, 1986. doi: 10.1109/ROBOT.1986.1087395.

[50] S. C. Jacobsen, H. Ko, E. K. Iversen, and C. C. Davis. Antagonistic control of a tendon driven manipulator. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 1334–1339, 1989. doi: 10.1109/ROBOT.1989.100165.

[51] M. Jäntsch, S. Wittmeier, and A. Knoll. Distributed control for an anthropomimetic robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 5466–5471, Oct. 2010. doi: 10.1109/IROS.2010.5651169.

[52] M. Jäntsch, C. Schmaler, S. Wittmeier, K. Dalamagkidis, and A. Knoll. A scalable Joint-Space Controller for Musculoskeletal Robots with Spherical Joints. In *Proc. IEEE International Conference on Robotics and Biomimetics ROBIO*, pages 2211–2216, Dec. 2011. doi: 10.1109/ROBIO.2011.6181620.

[53] M. Jäntsch, S. Wittmeier, K. Dalamagkidis, and A. Knoll. Computed Muscle Control for an Anthropomimetic Elbow Joint. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 2192–2197, Oct. 2012. doi: 10.1109/IROS.2012.6385851.

[54] M. Jäntsch, S. Wittmeier, K. Dalamagkidis, A. Panos, F. Volkart, and A. Knoll. Anthrob – A Printed Anthropomimetic Robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[55] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear estimation*. Prentice Hall New Jersey, 2000.

[56] E. R. Kandel, J. H. Schwartz, and T. M. Jessel. *Principles of Neural Science*. McGraw-Hill, 4 edition, 2000.

[57] D. Karnopp. Computer simulation of stick-slip friction in mechanical dynamic systems. *Journal of Dynamic Systems, Measurement, and Control*, 107:100, 1985.

[58] S. Kawamura, H. Kino, and C. Won. High-speed manipulation by using parallel wire-driven robots. *Robotica*, 18(1):13–21, Jan. 2000. ISSN 0263-5747. doi: 10.1017/S0263574799002477.

[59] M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, Dec. 1999.

[60] H. K. Khalil. *Nonlinear systems*, volume 3. Prentice Hall New Jersey, 3 edition, 2002.

[61] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, 36(1):42–59, Apr. 2012. ISSN 13675788. doi: 10.1016/j.arcontrol.2012.03.004.

[62] H. Kino, S. Kikuchi, T. Yahiro, and K. Tahara. Basic study of biarticular muscle's effect on muscular internal force control based on physiological hypotheses. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 4195–4200, 2009. doi: 10.1109/ROBOT.2009.5152196.

[63] H. Kobayashi and R. Ozawa. Adaptive neural network control of tendon-driven mechanisms with elastic tendons. *Automatica*, 39(9):1509–1519, 2003.

[64] H. Kobayashi, K. Hyodo, and D. Ogane. On Tendon-Driven Robotic Mechanisms with Redundant Tendons. *The International Journal of Robotics Research*, 17(5):561–571, May 1998. doi: 10.1177/027836499801700507.

[65] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84:171–203, Nov. 2010. ISSN 0885-6125. doi: 10.1007/s10994-010-5223-6.

[66] K. Koganezawa and H. Yamashita. Stiffness control of multi-DOF joint. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 363–370. IEEE, Oct. 2009. ISBN 978-1-4244-3803-7. doi: 10.1109/IROS.2009.5354434.

[67] T. Kozuki, H. Mizoguchi, Y. Asano, M. Osada, T. Shirai, U. Junichi, Y. Nakanishi, K. Okada, and M. Inaba. Design Methodology for the Thorax and Shoulder of Human Mimetic Musculoskeletal Humanoid Kenshiro -A Thorax structure with Rib like Surface -. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 3687–3692, 2012. ISBN 9781467317351.

[68] B. C. Kuo and M. F. Golnaraghi. *Automatic control systems*, volume 4. John Wiley & Sons New York, 9 edition, 2009.

[69] C. L. Lawson and R. J. Hanson. *Solving least squares problems*. Society for Industrial and Applied Mathematics, 3 edition, 1995. ISBN 0-7695-2288-2.

[70] L. Le-Tien and A. Albu-Schäffer. Adaptive Friction Compensation in Trajectory Tracking Control of DLR Medical Robots with Elastic Joints. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 1149–1154, 2012. ISBN 9781467317351.

[71] Y. LeCun, L. Bottou, and G. B. Orr. Efficient BackProp. In *Neural networks: Tricks of the trade*, pages 9–50. Springer Berlin Heidelberg, 1998.

[72] Y.-t. Lee, H.-r. Choi, W.-k. Chung, and Y. Youm. Stiffness Control of a Coupled Tendon-Driven Robot Hand. *IEEE Control Systems Magazine*, 14:10–19, 1994.

[73] T. Lens and O. von Stryk. Investigation of Safety in Human-Robot-Interaction for a Series Elastic, Tendon-Driven Robot Arm. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 4309–4314, 2012. ISBN 9781467317351.

[74] H. Liu, K. Wu, P. Meusel, N. Seitz, G. Hirzinger, M. H. Jin, Y. W. Liu, S. W. Fan, T. Lan, and Z. P. Chen. Multisensory five-finger dexterous hand: The DLR/HIT Hand II. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 3692–3697, 2008. doi: 10.1109/IROS.2008.4650624.

[75] H. G. Marques, R. Newcombe, and O. Holland. Controlling an Anthropomimetic Robot: A Preliminary Investigation. In *Lecture Notes in Computer Science*, volume 464, pages 736–745. 2007.

[76] H. G. Marques, M. Jäntsch, S. Wittmeier, O. Holland, C. Alessandro, A. Diamond, M. Lungarella, and R. Knight. ECCE1: The first of a series of anthropomimetic musculoskeletal upper torsos. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 391–396, 2010. doi: 10.1109/ICHR.2010.5686344.

[77] C. Mayhew, R. Sanfelice, and A. Teel. Quaternion-Based Hybrid Control for Robust Global Attitude Tracking. *IEEE Transactions on Automatic Control*, 56(11):2555–2566, 2011. ISSN 0018-9286. doi: 10.1109/TAC.2011.2108490.

[78] J. L. Meriam and L. G. Kraige. *Engineering Mechanics: Statics*. Wiley, 5 edition, 2002.

[79] J. L. Meriam and L. G. Kraige. *Engineering Mechanics: Dynamics SI Version*, volume 2. Wiley, 6 edition, 2008.

[80] S. A. Migliore, E. A. Brown, and S. P. DeWeerth. Biologically Inspired Joint Stiffness Control. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 4508– 4513, Apr. 2005. ISBN 0-7803-8914-X. doi: 10.1109/ROBOT.2005.1570814.

[81] D. Mitrovic, S. Klanke, and S. Vijayakumar. Learning Impedance Control of Antagonistic Systems Based on Stochastic Optimization Principles. *The International Journal of Robotics Research*, 2010. doi: 10.1177/0278364910387653.

[82] D. Mitrovic, S. Klanke, and S. Vijayakumar. Adaptive optimal feedback control with learned internal dynamics models. In O. Sigaud and J. Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264, pages 65–84. Springer Berlin Heidelberg, 2010.

[83] I. Mizuuchi, R. Tajima, T. Yoshikai, D. Sato, K. Nagashima, M. Inaba, Y. Kuniyoshi, and H. Inoue. The Design and Control of the Flexible Spine of a Fully Tendon-Driven Humanoid "Kenta". In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 2527–2532, 2002.

[84] I. Mizuuchi, T. Yoshikai, Y. Sodeyama, Y. Nakanishi, A. Miyadera, T. Yamamoto, T. Niemelä, M. Hayashi, J. Urata, Y. Namiki, T. Nishino, and M. Inaba. Development of musculoskeletal humanoid Kotaro. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 82–87, 2006. doi: 10.1109/ROBOT.2006.1641165.

[85] I. Mizuuchi, Y. Nakanishi, Y. Sodeyama, Y. Namiki, T. Nishino, N. Muramatsu, J. Urata, K. Hongo, T. Yoshikai, and M. Inaba. An advanced musculoskeletal humanoid Kojiro. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 294–299, 2007. doi: 10.1109/ICHR.2007.4813883.

[86] A. Morecki, Z. Busko, H. Gasztold, and K. Jaworek. Synthesis and control of the anthropomorphic two-handed manipulator. In *Proc. 10th International Symposium on Industrial Robots*, pages 71–77, 1980.

[87] J. Na, X. Ren, G. Herrmann, and Z. Qiao. Adaptive neural dynamic surface control for servo systems with unknown dead-zone. *Control Engineering Practice*, 19(11): 1328–1343, Nov. 2011. ISSN 0967-0661. doi: 10.1016/j.conengprac.2011.07.005.

[88] Y. Nakanishi, I. Mizuuchi, T. Yoshikai, T. Inamura, and M. Inaba. Pedaling by a redundant musculo-skeletal humanoid robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 68–73, Dec. 2005. ISBN 0-7803-9320-1. doi: 10.1109/ICHR.2005.1573547.

[89] Y. Nakanishi, K. Hongo, I. Mizuuchi, and M. Inaba. Joint Proprioception Acquisition Strategy Based on Joints-Muscles Topological Maps for Musculoskeletal Humanoids. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 1727–1732, 2010.

[90] Y. Nakanishi, T. Izawa, M. Osada, N. Ito, S. Ohta, J. Urata, and M. Inaba. Development of Musculoskeletal Humanoid Kenzoh with Mechanical Compliance Changeable Tendons by Nonlinear Spring Unit. In *Proc. IEEE International Conference on Robotics and Biomimetics ROBIO*, pages 2384–2389, Dec. 2011. doi: 10.1109/ROBIO.2011.6181655.

[91] S. Nicosia and P. Tomei. A method to design adaptive controllers for flexible joint robots. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 701–706, 1992. doi: 10.1109/ROBOT.1992.220286.

[92] R. Niiyama, S. Nishikawa, and Y. Kuniyoshi. Athlete Robot with applied human muscle activation patterns for bipedal running. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 498–503. doi: 10.1109/ICHR.2010.5686316.

[93] R. Niiyama, A. Nagakubo, and Y. Kuniyoshi. Mowgli: A Bipedal Jumping and Landing Robot with an Artificial Musculoskeletal System. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 2546–2551, 2007. doi: 10.1109/ROBOT.2007.363848.

[94] R. Niiyama, S. Nishikawa, and Y. Kuniyoshi. Biomechanical Approach to Open-Loop Bipedal Running with a Musculoskeletal Athlete Robot. *Advanced Robotics*, 26 (3-4):383–398, 2012.

[95] J. H. Oh and J. S. Lee. Control of flexible joint robot system by backstepping design approach. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, volume 4, pages 3435–3440, Apr. 1997. doi: 10.1109/ROBOT.1997.606867.

[96] H. Olsson, K. J. Å ström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky. Friction Models and Friction Compensation. *European Journal of Control*, 4:176–195, 1998.

[97] C. Ott, A. Albu-Schäffer, A. Kugi, and G. Hirzinger. On the Passivity-Based Impedance Control of Flexible Joint Robots. *IEEE Transactions on Robotics*, 24(2): 416–429, 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.915438.

[98] G. Palli. *Model and Control of Tendon Actuated Robots*. PhD thesis, Università degli Studi di Bologna, 2006.

[99] G. Palli, C. Melchiorri, T. Wimbock, M. Grebenstein, and G. Hirzinger. Feedback linearization and simultaneous stiffness-position control of robots with antagonistic actuated joints. *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 4367–4372, Apr. 2007. ISSN 1050-4729. doi: 10.1109/ROBOT.2007.364152.

[100] C. Pelchen, C. Schweiger, and M. Otter. Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes. In *2nd International Modelica Conference*, pages 257–266, 2002.

[101] J. Peters and S. Schaal. 2008 Special Issue: Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2008.02.003.

[102] R. Pfeifer, F. Iida, and J. C. Bongard. New Robotics: Design Principles for Intelligent Systems. *Artificial Life*, 11(1-2):99–120, 2005. ISSN 1064-5462. doi: http://dx.doi.org/10.1162/1064546053279017.

[103] R. Pfeifer, M. Lungarella, and F. Iida. Self-Organization, Embodiment, and Biologically Inspired Robotics. *Science*, 318(5853):1088–1093, Nov. 2007.

[104] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical report, MIT Artificial Intelligence Laboratory, Cambridge, MA, USA, 1989.

[105] V. Potkonjak, B. Svetozarevic, K. Jovanovic, and O. Holland. Control of Compliant Anthropomimetic Robot Joint. In *Symposium on Computational Geometric Methods in Multibody System Dynamics*, pages 1271–1274, 2010.

[106] V. Potkonjak, B. Svetozarevic, K. Jovanovic, and O. Holland. Biologically-inspired control of a compliant anthropomimetic robot. In *Symposium on Computational Geometric Methods in Multibody System Dynamics*, pages 1271–1274, 2010.

[107] K. Radkhah, T. Lens, and O. von Stryk. Detailed dynamics modeling of BioBiped's monoarticular and biarticular tendon-driven actuation system. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 4243–4250, Oct. 2012. ISBN 978-1-4673-1736-8. doi: 10.1109/IROS.2012.6385564.

[108] J. Reisinger and A. Steininger. The design of a fail-silent processing node for the predictable hard real-time system MARS. *Distributed Systems Engineering*, 1(2):104–111, 1993.

[109] M. Rickert. Robotics Library (RL) v0.6, 2012. URL http://sourceforge.net/projects/roblib/.

[110] P. Rochat. Self-perception and action in infancy. *Experimental Brain Research*, 123 (1-2):102–109, Oct. 1998. ISSN 0014-4819, 1432-1106. doi: 10.1007/s002210050550.

[111] A. Rost and A. Verl. The QuadHelix-Drive - An improved rope actuator for robotic applications. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 3254–3259, 2010. doi: 10.1109/ROBOT.2010.5509764.

[112] D. E. Rumelhart, G. E. Hintont, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[113] J. K. Salisbury and J. J. Craig. Articulated Hands - Force Control And Kinematic Issues. *The International Journal of Robotics Research*, 1(1):4–17, Mar. 1982. ISSN 02783649. doi: 10.1177/027836498200100102.

[114] C. Schmaler. *Extracting the Muscle Jacobian for Anthropomimetic Robot Control using Machine Learning*. Master thesis, Technische Universität München, 2011.

[115] A. L. Schwab and J. P. Meijaard. How to draw Euler angles and utilize Euler parameters. In *Proceedings of IDETC/CIE*, 2008.

[116] K. Severinson-Eklundh, A. Green, and H. Hüttenrauch. Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, 42(3-4): 223–234, Mar. 2003. ISSN 0921-8890. doi: 10.1016/S0921-8890(02)00377-9.

[117] Shadow. Shadow Dexterous Hand E1 Series, Jan. 2013. URL http://www.shadowrobot.com/wp-content/uploads/shadow_dexterous_hand_technical_specification_E1_20130101.pdf.

[118] K. Shoemake. Animating rotation with quaternion curves. *Proc. ACM SIGGRAPH Computer Graphics*, 19(3):245–254, July 1985. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/325165.325242.

[119] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer New York, 2007. ISBN 354023957X.

[120] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics - Modelling, Planning and Control*. Springer, 2009.

[121] J.-J. E. Slotine and W. Li. Adaptive manipulator control: A case study. *IEEE Transactions on Automatic Control*, 33(11):995–1003, Nov. 1988. ISSN 0018-9286. doi: 10.1109/9.14411.

[122] M. W. Spong. Adaptive control of flexible joint manipulators: Comments on two papers. *Automatica*, 31(4):585–590, Apr. 1995. ISSN 0005-1098. doi: 10.1016/0005-1098(95)98487-Q.

[123] D. G. Steele. *The anatomy and biology of the human skeleton*. Texas A&M University Press, 1988. ISBN 0890963266.

[124] R. Stribeck and M. Schröter. *Die wesentlichen Eigenschaften der Gleit-und Rollenlager: Untersuchung einer Tandem-Verbundmaschine von 1000 PS*. Springer, 1903.

[125] D. Swaroop, J. K. Hedrick, P. P. Yip, and J. C. Gerdes. Dynamic surface control for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 45(10):1893–1899, 2000.

[126] K. Tahara and H. Kino. Iterative learning control for a redundant musculoskeletal arm: Acquisition of adequate internal force. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 234–240, 2010. doi: 10.1109/IROS.2010.5650071.

[127] K. Tahara and H. Kino. Iterative Learning Scheme for a Redundant Musculoskeletal Arm: Task Space Learning with Joint and Muscle Redundancies. In *Proc. International Conference on Broadband, Wireless Computing, Communication and Applications BWCCA*, pages 760–765, 2010. doi: 10.1109/BWCCA.2010.168.

[128] K. Tahara, S. Arimoto, M. Sekimoto, and Z.-W. Luo. On control of reaching movements for musculo-skeletal redundant arm model. *Applied Bionics and Biomechanics*, 6(1):11–26, 2009. ISSN 1176-2322.

[129] K. Tahara, Y. Kuboyama, and R. Kurazume. Iterative learning control for a musculoskeletal arm: Utilizing multiple space variables to improve the robustness. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 4620–4625, Oct. 2012. ISBN 978-1-4673-1736-8. doi: 10.1109/IROS.2012.6385628.

[130] D. G. Thelen and F. C. Anderson. Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. *Journal of Biomechanics*, 39(6):1107–1115, 2006. ISSN 0021-9290. doi: 10.1016/j.jbiomech.2005.02.010.

[131] D. G. Thelen, F. C. Anderson, and S. L. Delp. Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics*, 36(3):321–328, 2003. ISSN 0021-9290. doi: DOI:10.1016/S0021-9290(02)00432-3.

[132] L. Tian and A. A. Goldenberg. Robust adaptive control of flexible joint robots with joint torque feedback. In *Proc. IEEE International Conference on Robotics and Automation ICRA*, pages 1229–1234, 1995. doi: 10.1109/ROBOT.1995.525448.

[133] C. Tin and C.-S. Poon. Internal models in sensorimotor integration: perspectives from adaptive control theory. *Journal of Neural Engineering*, 2(3):147–163, Sept. 2005. doi: 10.1088/1741-2560/2/3/S01.

[134] G. J. van Ingen Schenau, M. F. Bobbert, and R. H. Rozendal. The unique action of bi-articular muscles in complex movements. *Journal of Anatomy*, 155:1–5, Dec. 1987. ISSN 0021-8782.

[135] S. N. Vukosavic. Modeling and Supplying DC Machines. In *Electrical Machines*, Power Electronics and Power Systems, chapter 11, pages 299–342. Springer New York, Jan. 2013. ISBN 978-1-4614-0399-9, 978-1-4614-0400-2.

[136] D. Wang and J. Huang. Neural network-based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict-feedback form. *IEEE Transactions on Neural Networks*, 16(1):195–202, 2005. ISSN 1045-9227. doi: 10.1109/TNN.2004.839354.

[137] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

[138] J. S. Wilson. *Sensor technology handbook*. Newnes, 2004.

[139] S. Wittmeier. *Physics-Based Modeling and Simulation of Musculoskeletal Robots [in preparation]*. Phd thesis, Technische Universität München, 2013.

[140] S. Wittmeier, M. Jäntsch, K. Dalamagkidis, and A. Knoll. Physics-based Modeling of an Anthropomimetic Robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 4148–4153, Oct. 2011. doi: 10.1109/IROS.2011.6094459.

[141] S. Wittmeier, M. Jäntsch, K. Dalamagkidis, M. Rickert, H. G. Marques, and A. Knoll. CALIPER: A Universal Robot Simulation Framework For Tendon-Driven Robots. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 1063–1068, Oct. 2011. doi: 10.1109/IROS.2011.6094455.

[142] S. Wittmeier, A. Gaschler, M. Jäntsch, K. Dalamagkidis, and A. Knoll. Calibration of a physics-based model of an anthropomimetic robot using evolution strategies. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 445–450, Oct. 2012. doi: 10.1109/IROS.2012.6385591.

[143] S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntsch, K. Jovanovic, R. Knight, and H. G. Marques. Towards Anthropomimetic Robotics: Development, Simulation, and Control of a Musculoskeletal Torso. *Artificial Life*, 19(1):171–193, 2013.

[144] P. Wolfe. The simplex method for quadratic programming. *Econometrica: Journal of the Econometric Society*, 27(3):382–398, 1959.

[145] B. Xian, M. S. de Queiroz, D. Dawson, and I. Walker. Task-space tracking control of robot manipulators via quaternion feedback. *IEEE Transactions on Robotics and Automation*, 20(1):160–167, 2004. doi: 10.1109/TRA.2003.820932.

[146] K. Yamane and Y. Nakamura. Robot Kinematics and Dynamics for Modeling the Human Body. In *Springer Tracks in Advanced Robotics*, volume 66, pages 49–60. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14742-5, 978-3-642-14743-2.

[147] S. J. Yoo, J.-B. Park, and Y.-H. Choi. Adaptive Dynamic Surface Control for Stabilization of Parametric Strict-Feedback Nonlinear Systems With Unknown Time Delays. *IEEE Transactions on Automatic Control*, 52(12):2360–2365, 2007. ISSN 0018-9286. doi: 10.1109/TAC.2007.910715.

[148] W. H. Young. On Classes of Summable Functions and their Fourier Series. *Proceedings of the Royal Society of London. Series A*, 87(594):225–229, Aug. 1912. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.1912.0076.

[149] V. M. Zatsiorsky. *Kinetics of Human Motion*. Human Kinetics Publishers, 2002.