Universität Karlsruhe (TH)
Institut für Industrielle
Anwendungen der Informatik
und Mikrosystemtechnik

Position Control of a Humanoid Robot Arm actuated by Artificial Muscles

Diploma Thesis

Joachim Schröder

December 1st, 2002 – May 31st, 2003

Supervisor: Prof. Dr.-Ing. R. Dillmann
Co-Supervisor: Prof. Dr. K. Kawamura
Advisor: Dipl.-Ing. T. Gockel

Declaration:

I hereby declare that this thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration unless stated otherwise.

Karlsruhe, May 31st 2003

Joachim Schröder

Acknowledgments

This research is conducted with financial support of the Center for Intelligent Systems (CIS) and the Institute for Industrial Applications of Informatics and Microsystems (IAIM). The research is subject of my Diploma Thesis at the University of Karlsruhe (TH), written as a visiting scholar at Vanderbilt University.

First, I want to thank Dr. Kawamura and Prof. Dillmann who made this all possible by establishing the contact and supporting me financially.

I thank my advisor in Germany Tilo Gockel for his advice and help even across the Atlantic Ocean.

I want to thank my colleagues at the Intelligent Robotics Lab, especially Flo Fottrell who had to deal with a lot of paperwork before and during my stay, Dr. Juyi Park for discussions and answering questions, Dr. Bugra Koku and Jane Wilkes for taking care of me during the first weeks and helping me to gain ground.

Special thanks to my mother and my father, who always accepted the decisions I made and helped me in achieving my aims.

I want to thank all my friends and all people who supported me in any way and made my stay in the United States a great experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the Center for Intelligent Robotics at Vanderbilt University is one emphasis the development and control of humanoid robots. One of the current projects is the development of an Intelligent Soft Arm Control, called ISAC. This is an intelligent robotics aid system for the service sector and to be used in hospitals and at home. The main benefit of such a system is to provide the sick and physically challenged person with means to live independently. A imaginable example is shown in Figure 1.1.

Because of the direct contact to humans, the robot has to fulfill a lot more safety precautions concerning its movements and power than an industrial robot. To guarantee the safe interaction between robot and humans, and at the same time to achieve a more natural movement, the so called Soft Arm is driven by pneumatic actuators. These actuators, also known as McKibben muscles or artificial muscles, resemble in a high manner the movements of the human muscle. They are lightweight, safe to operate and have therefore a high potential to be used as human aid in the service sector.
The problem by using this kind of actuators is that the muscles are highly nonlinear and contain a hysteresis. This is the reason why controlling these actuators is not simple and a robust control architecture is needed to handle that task.

In the past, lots of scientists have developed models to describe the behavior of artificial muscles. The development of control architectures based on conventional principles such as PID, and also Fuzzy and Neural Network controls have been subject of several research papers.

In the course of this work, the existing models are introduced and the previous work on artificial muscle control is discussed. A convenient muscle model was chosen and extended. Furthermore, a position control for the Soft Arm is developed in theory, implemented on the system and verified in experiments.

1.2 Structure of the report

In the second chapter, basics of artificial muscles and different kinds of actuators are presented. The humanoid robot system ISAC is introduced. A description of previous research, existing muscle models and control approaches follows.

An actuator model is developed in chapter three. On basis of the existing models, the most suitable is picked out, extended and customized for our robot and finally verified on a testbed.

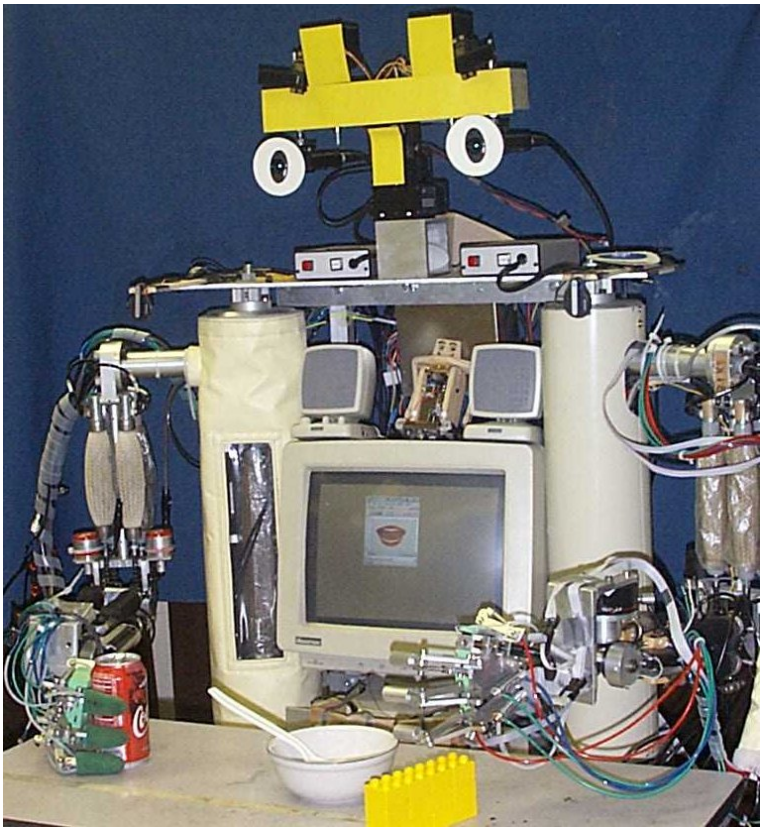

Figure 1.1: Application in the human aid sector

In the fourth chapter, models for the system components are derived. This includes building transfer functions for hardware components as well as developing a kinematic model of the Soft Arm and solving direct and inverse kinematics.

Chapter five describes the development of the control architecture. Different architectures were implemented on a testbed, experiments were run and a final control strategy for ISAC was chosen.

The sixth chapter deals with the control implementation on the real Soft Arm.

Chapter seven presents experimental results that were run after implementing the control architecture on ISAC.

In the last chapter, a conclusion is drawn and possible future tasks are addressed and presented.

Descriptions and specifications of all used hardware components can be found in the appendix. The software controlling ISAC is described, and additional hardware documentation can be found here.

Chapter 2

Basics

2.1 Artificial muscles

To receive an overlook over different artificial muscle types, this chapter will list them and explain the main differences. Furthermore, the Intelligent Soft Arm Control (ISAC) is introduced and explained. At least, this chapter informs about the actual state of technology on the sector of artificial muscle modeling and control.

2.1.1 Functionality of artificial muscles

The artificial muscles used in this work are pneumatic actuators. They consist of an outer cord netting and a inner rubber tube. The inner tube is expanding when inflated, and the cord which is not expandable transfers this change of volume into a change of length.

Figure 2.1: Structure of artificial muscles [9]

The compressible air makes the muscles elastic and gives them spring-like characteristics. A robot arm actuated by artificial muscles has therefore lower stiffness than e.g. a joint actuated by electrical motors. These properties make artificial muscles safe to operate in a human environment and thus interesting for the direct robot-human interaction. Artificial muscles are well suited for a number of robotic applications because they are lightweight, naturally compliant, and have a high force-to-weight ratio [14]. Construction and Materials differ from manufacturer to manufacturer, but the working principle is always the same.

The static state of artificial muscles can be described by the following variables:

- Internal pressure p of the muscle
- Contraction force F
- Contraction ratio ϵ, some formulas use instead of the contraction ratio the angle Θ between an elemental length of the helical fiber and the longitudinal axis of the muscle

While artificial muscles have many desirable properties, they exhibit also some disadvantages. Due to friction between the outer cord and the inner rubber tube, the relation between pressure, contraction and force contains a hysteresis. The resulting complex behavior requires accurate muscle modeling and a robust control technique [21].

Two of the muscles can be combined to one rotational actuator. Such an actuator works the same way a human joint does, one of the muscles is representing the agonist (the working or active muscle) and the other one is the antagonist (the counter part to decelerate and stabilize the joint). Figure 2.2 shows the composition of the muscles in the artificial joint. The variables, describing the state of the joint are:

- The differential pressure $\Delta p = \frac{1}{2}(p_2 - p_1)$
- The torque T of the joint, where $T = (F_2 - F_1)r$
- The angle φ of the joint, which is computed out of $\varphi = \frac{l_2 - l_1}{r}$

The expression $\Delta p = \frac{1}{2}(p_2 - p_1)$ is used in this context because the variable Δp usually describes in many papers and formulas the difference between the initial pressure p_0 and the muscle pressure as shown in the next paragraph.

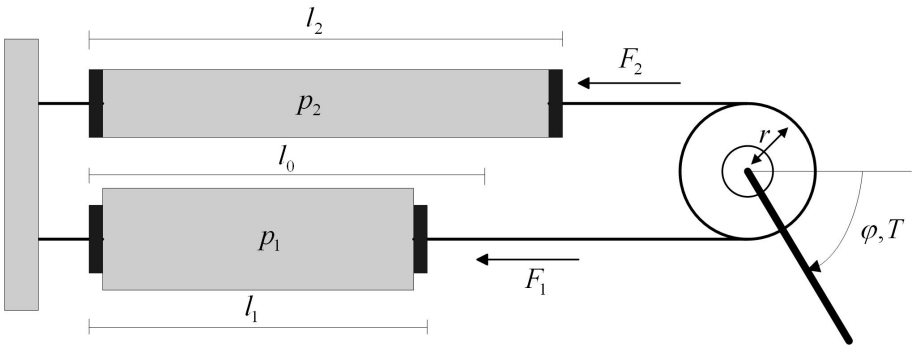

Figure 2.2: Two muscles combined to one rotational actuator

The joint is generally activated by increasing and decreasing the muscles pressure upon the same amount. That means, the muscles are inflated to an initial pressure p_0, and a torque is applied by giving $p_1 = p_0 - \Delta p$ and $p_2 = p_0 + \Delta p$. This is the way muscles are activated in most of the papers. There are other imaginable possibilities, like keeping one muscle always at highest pressure and changing only the pressure of the other one to move the joint. To change direction, reverse the procedure.

The first kind of symmetric muscle activation has proved to be suitable in the past and will be used in the experiments. The advantage of this procedure is that while one muscle shortens, the second one is lengthening approximately the same amount. This guarantees a high stiffness for the entire workspace of the joint and helps holding the chain on the wheels so that additional mechanic springs to strain the chain are needless.

2.1.2 History of artificial muscles

Artificial muscles have first been developed in 1957 by the physician Joseph L. McKibben, who developed an artificial actuator to aid in improving of his daughter's grasping ability [17]. Since then, artificial muscles based on the principle of inflatable tubes are also known as McKibben muscles. The muscles were originally intended in the medical sector to actuate artificial limbs. Today, they are usually used in robotics.

In the 1980's, the Bridgestone company commercialized the product for industry and distributed the muscles as so called Rubbertuators. Later, production of Rubbertuators has been stopped because Bridgestone backed out of robotics. Since then, the Shadow Robot Company in England is manufacturing artificial muscles for robotic applications. The German Festo Company is distributing artificial muscles for use in industrial environments.

2.1.3 McKibben and Rubbertuator muscles

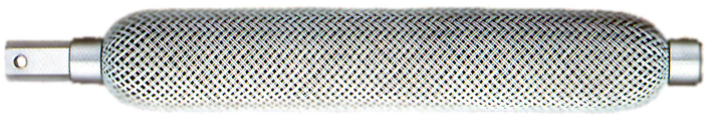

Figure 2.3: Bridgestone Rubbertuator

The first artificial muscle developed by McKibben consisted of an inner rubber tube and an outer helical weave tube. The muscle was sealed at its ends, and when pressurized, it expanded in radial direction but contracted axially at the same time. Figure 2.4 shows the layers and extreme states of a McKibben muscle.

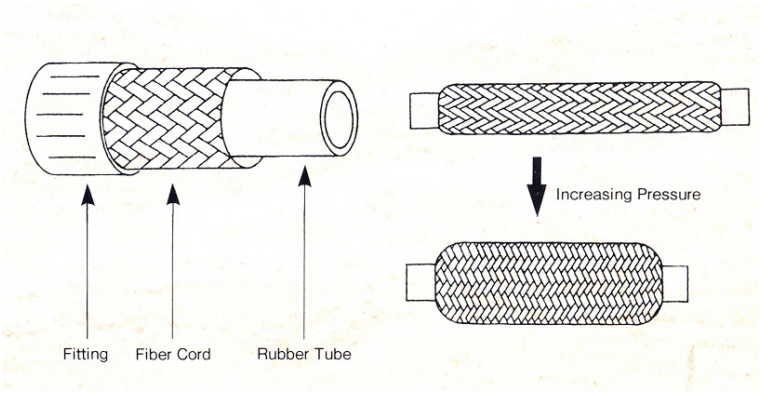

Figure 2.4: Rubbertuator construction [7]

.

In the 1960's, electric motors replaced the McKibben muscle as an orthotic device, because of the smaller design and the easier way to control. The interest in artificial muscles increased when the Bridgestone Company redesigned the McKibben muscle and distributed them as Rubbertuators.

In the course of this development, Bridgestone also designed robots driven by Rubbertuators, such as the Soft Arm used in the humanoid robot system at Vanderbilt University.

Several projects running at other research labs and universities show that the interest in artificial muscles for robot applications is still high and will still be subject of research in the next years.

2.1.4 Shadow muscles

In the 1980's, the Shadow Robot Company [9] in England set the goal to build a useful general-purpose robot. The first project was a walking robot, called the Shadow Biped. During the development of this robot, another version of the McKibben muscles for robot applications was invented. These muscles are based on the same principle but use a different fiber cord to reduce friction and hysteresis influence (Figure 2.5).

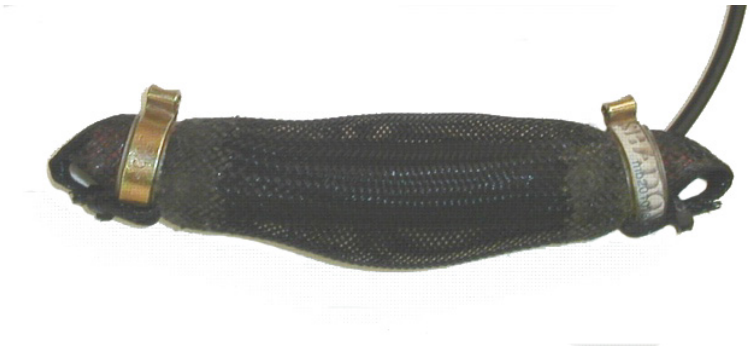

Figure 2.5: Shadow Air Muscle

The Shadow Company started lots of other robot projects like the Shadow Hand or the Zephyrus walking robot. A picture of the hand is shown in Figure 2.6.

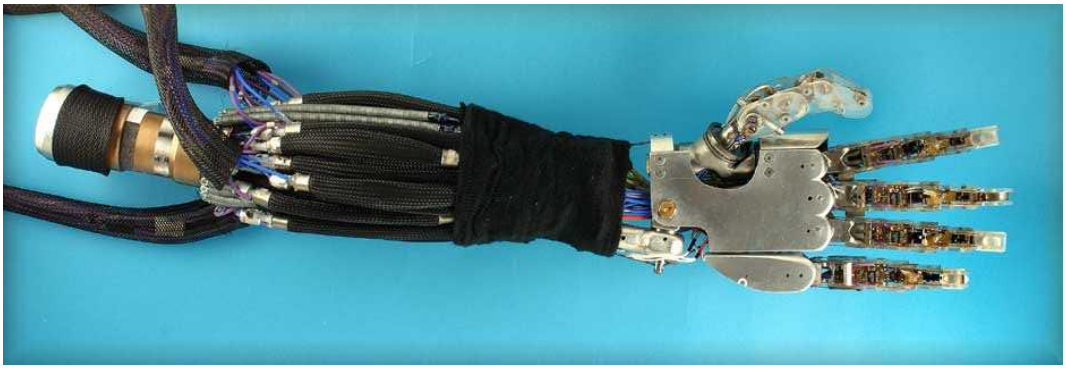

Figure 2.6: Shadow Hand

2.1.5 Festo muscles

The Festo Company built a muscle for industrial use, it is shown in Figure 2.7. This product, distributed as "Fluidic Muscle", has a different construction than Shadow or Bridgestone muscles. It consists of several layers, all these layers contain a membrane in which the netting and the rubber tube are combined.
In the first place, this construction prohibits that the net layers are rubbing on each other, so the life cycle is increased. In the second place, the friction which is the main cause for the hysteresis is reduced through separating the netting layers. The consequences are much better muscle properties, concerning hysteresis and nonlinearity.

First developed for industrial applications, Festo muscles were later used in humanoid robotics at the University of Karlsruhe (TH) [19]. Their most famous project based on artificial muscles is the six-legged walking machine Airbug [20].

Figure 2.7: Festo Fluidic Muscles

2.2 The humanoid robot system ISAC

The Intelligent Soft Arm Control (ISAC, Figure 2.8) has been developed at the Center for Intelligent Systems. ISAC consists of a human-like upper body with two six degree-of-freedom serial robot manipulators, called Soft Arms [7]. Each Soft Arm is a pneumatically driven robotic manipulator that is actuated by artificial muscles.

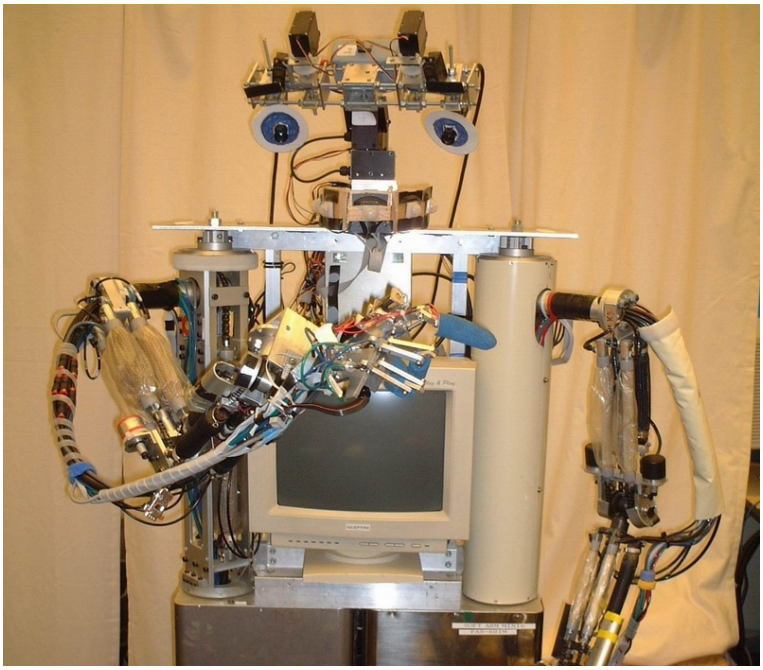

Figure 2.8: The humanoid robot system ISAC

At this time, Bridgestone Rubbertuators are used, but they will be replaced by Shadow muscles or others in future because the supply is limited. Therefore, experiments and modeling is primarily done with Shadow muscles. Optical encoders on each of ISAC's joints provide information about the actual joint angle.

2.3 Actual state of technology

To provide an insight into modeling and control of artificial muscles, this section describes the actual state of technology. The different existing models are introduced, and discussed concerning the fit for our needs. A lot of control strategies have been developed to control artificial muscles and have given partially very good results, but some of them seem not useful to be implemented on ISAC. This section judges the applicability of these strategies for our work.

2.3.1 Pneumatic actuator modeling

The physical muscle model describes the behavior of the muscles, which means the relation between given pressure, actual position and resulting force. This information can easily be expanded to describe the joints behavior with delta pressure, actual angle and resulting torque.

Finding a physical model for Rubbertuators has been subject of many papers and publications [5, 6, 13, 23, 22]. That is the reason why it is not the goal to derive another model, but to compare the existing models and find the best suitable for the Soft Arm. The model can be modified in different ways to customize it for our needs, but the major task in this thesis is the arm control, not artificial muscle modeling.

Most of the existing models are physical static models. The aim of deriving these models was to find a relation between pressure, contraction and force of the muscle. In most cases the models were found over the approach on the view of energy conservation. According to [17], Schulte was the first who published a model in 1961. When Bridgestone started manufacturing Rubbertuators, they presented a static model that described the behavior of artificial muscles, also based on this principle.

Chou and Hannaford from the University of Washington presented another model in 1994 [5]. They showed a second model in the same report, which considered the wall thickness of shell and bladder. The first of these models is probably the most famous model, and a lot of fractionally different models used this one as a basis. Therefore, the derivation of this model is shown on the next page, representing all models based on the principle of the view of energy conservation.

Other models have been presented using the Chou-Hannaford-Model or the Bridgestone model as basis, so for example the model of Cai and Yamaura [1]. They added a part describing the stickiness and created so a dynamical model. Tsagarakis and Caldwell published an improved model in 2000 [22], in which they considered distortion effects at the termination nodes and radial pressure loss due to radial elasticity. Kerscher [15] considered in his model the fact that the muscle is not a cylinder in the end nodes. He added a term to the contraction which contains an exponential function to compensate this error.

In 1998, Repperger tried to find a second order nonlinear differential equation. He did not consider the geometry in his equation, this is done by designating the parameters through a system identification.

Another way to find a muscle model is to use a polynomial equation like Caldwell, Cerda and Goodwin did in their research paper, published in 1995 [3]. The problem with a pure mathematical model is, that it cannot be reacted easily on parameter changes, or on changes of environmental conditions. For example, a polynomial model contains some coefficients that have to be identified. If there are no parameters like muscle diameter present in the equation, the coefficients are useless for other muscles with different diameters or in different environments. Having a physical representation of the muscle would help to react on changing parameters or draw conclusions on other types of muscles.

In the next section, the best known models are presented in detail. All models are physical models, static or dynamic and are based on the view of energy conservation.

The Schulte model

According to [17], Schulte published a physical model of the actuator in 1961. He simplified the unpublished equation of McKibben's original formula, which included several nonlinear terms containing friction and effects of tubing elasticity. By deriving a simplified form of this equation, he came to the following result:

$$F = \frac{\pi D^2 p}{4}(3cos^2(\Theta) - 1) \tag{2.1}$$

Where:

F	:	actuator force
p	:	internal pressure
θ	:	angle between an elemental length of the helical fiber and the longitudinal axis of the actuator
D	:	diameter of actuator when θ is 90 degrees

The Bridgestone model

This model was published by the manufacturer in 1980. The model has the following form:

$$F = {D_0}^2 p[a(1-\epsilon)^2 - b] \tag{2.2}$$

Where:

F	:	actuator force
p	:	internal pressure
ϵ	:	contraction ratio
a,b	:	inherent Rubbertuator constants
D_0	:	effective diameter before displacement

As can be seen in this equation, the contraction force does not only depend on the actual pressure, it also depends on the contraction ratio. This explains the spring-like behavior of Rubbertuators.
Bridgestone gives the advice [7] to apply pressures $p_1 = p_0 - \Delta p$ and $p_2 = p_0 + \Delta p$ when using the muscles in rotational joint structure. According to [7], the spring constant of the joint can be changed by changing the initial pressure p_0.

This model given by Bridgestone seems also very similar to the one from Schulte. At a more in depth view, it turns out that the two models differ only in their coefficients. The Bridgestone model can be transformed into the Schulte model by choosing $b = \frac{\pi}{4}$ and $a = \pi\frac{3}{4}\frac{{l_0}^2}{b^2}$ and using the relations $l = b\cos\Theta$, $\epsilon = \frac{l_0 - l}{l_0}$ and $D = D_0$. The meaning of these values is described more precisely in the next section.

The Chou and Hannaford model

In their first paper about artificial muscle modeling in 1994, Chou and Hannaford presented a physical static Rubbertuator model [5]. A theoretical approach on energy conservation was the basis for this model. The advantage of this procedure is, that no detailed information

about the geometric muscle structure is needed. The only geometric variable existing in the formula is the length. Their approach in [5] is included here representative for all other models based on the principle of energy conservation.

The input work W_{in} is the work, needed to inflate the muscle, and the output work W_{out} is the work the muscle performs to the environment. Because of the limited ability of the muscle to store energy, input and output work can be assumed to be equal in this approach:

$$dW_{in} = dW_{out} \tag{2.3}$$

The internal work has to be applied through the internal pressure in the muscle, working against the rubber tube and can be described as:

$$dW_{in} = \int_{S_i} (p - p_0) \cdot dl_i \cdot ds_i = (p - p_0) \int_{S_i} dl_i \cdot ds_i = p \cdot dV \tag{2.4}$$

Where:

p	:	absolute internal gas pressure
p_0	:	environmental pressure
p	:	relative pressure $(p - p_0)$
S_i	:	inner surface
ds_i	:	surface vector
dl_i	:	inner surface displacement vector
dV	:	volume change of the muscle

Due to the fact that the muscle can only perform work by applying an axial force, the output work W_{out} is

$$dW_{out} = -F \cdot dl \tag{2.5}$$

where F is the axial force and l is the axial displacement. By inserting equations 2.3 and 2.4 in 2.5, the following relation is received:

$$F = -p\frac{dV}{dl} \tag{2.6}$$

Since the volume of the muscle can hardly be described, it is approximated by a perfect cylinder, given through the following formula:

$$V = \frac{1}{4}\pi D^2 l \tag{2.7}$$

Length l and Diameter D of the cylinder can be expressed by the equations

$$L = b\cos\Theta \text{ and } D = \tfrac{b\sin\Theta}{n\pi},$$

so that the cylinder volume changes to

$$V = \frac{b^3}{4\pi n^2}\sin^2\Theta\cos\Theta \tag{2.8}$$

where b is the thread length, Θ is the angle between a thread and the cylinder's longitudinal axis and n is the number of turns of one thread. The relationship between l, D, Θ and the constant parameters b and n is shown in Figure 2.9.

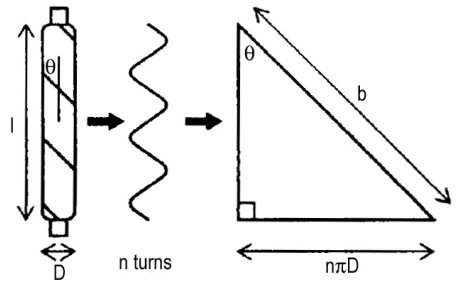

Figure 2.9: Relation between diameter D, muscle length l, thread angle Θ and the constant parameters b (thread length) and n (number of thread turns)

By inserting 2.8 in 2.6, the force can be expressed as

$$F = -p\frac{dV}{dL} = -p\frac{dV/d\Theta}{dl/d\Theta} = \frac{b^2(3\cos^2\Theta - 1)}{4\pi n^2} \tag{2.9}$$

Considering the fact that Θ is 90° in the rest position of the muscle, the diameter D_0 is $D_0 = b/n\pi$. This relation changes the formula to

$$F = \frac{\pi {D_0}^2 p}{4}(3\cos^2\Theta - 1) \tag{2.10}$$

which is the form published by Schulte in 1961.
In this derivation, the muscle volume was approximated with a cylinder. Instead of using equation 2.7, a smaller cylinder could be used, considering thickness of shell and bladder. If d describes the thickness, the following new form is given for the volume:

$$V = \frac{1}{4}\pi(D - 2d)^2 l \tag{2.11}$$

Inserting this new volume into 2.6, the force is now

$$F = \frac{\pi {D_0}^2 p}{4}(3\cos^2\Theta - 1) + p\pi[D_0 d(2\sin\Theta - \frac{1}{\sin\Theta}) - d^2] \tag{2.12}$$

After derivation of the muscle model, Chou and Hannaford performed some experiments. The following quasi-static and dynamic experiments were run with a constant pressure supply, by giving a slow triangular waveform displacement to the muscle. The frequency was first chosen to 1 Hz. The tension was recorded and showed a typical hysteresis. In a second experiment, they reduced the frequency to 0.25 Hz, but could not recognize a change of the hysteresis' form. Out of that, the authors concluded that the hysteresis is velocity independent. They explained this result with the fact that the velocity-independent Coulomb friction is much higher than the viscous friction, and therefore the main cause for the hysteresis.

In the next experiment, the operating range was reduced half, while keeping the frequency at 1 Hz. The result was that the rising paths of the hysteresis remained same, but the falling paths were different. The reason for that was the same initial condition, but another turning point. The authors followed that this experiment showed a history dependence of the hysteresis.

The Cai and Yamaura model

Cai and Yamaura presented a paper in 1996 [1], where they developed a dynamical model of a McKibben pneumatic muscle. Their goal was to realize a robust controller for an artificial muscle manipulator.

They used the following relation between pressure, force and contraction rate:

$$F = \alpha(1-\epsilon)^2 p + \beta p + \gamma \tag{2.13}$$

where α, β and γ are coefficients. This is nearly the same formula presented by Bridgestone, except the offset γ. To consider the stickiness of the muscle, a coefficient was added. This led to

$$F = \alpha(1-\epsilon)^2 p + \beta p - \omega\dot{\epsilon} + \gamma. \tag{2.14}$$

In opposite to the experiments Chou and Hannaford performed and to the conclusions they drew, the sticky part caused through the viscous friction was considered in this equation. Regrettably, the authors did not write what the reason for adding this part was and if experiments formed the basis for this decision. The results of their controller were good, but it is difficult to say whether the improved muscle model was the reason for that because the comparison with other models is missing. However, this is one way to describe the dynamic muscle behavior.

The Thilo Kerscher model

This model was developed by Kerscher in his Diploma Thesis in 2002 [15]. The subject of his thesis was the development of a controller for a walking machine, driven by artificial muscles. It is to say that he used Festo Fluidic Muscles, which have a different behavior than Rubbertuators. After the approach of energy conservation in the muscle, he received the following formula:

$$F = \pi {r_0}^2 p \cdot (a(1-\kappa)^2 - b) \tag{2.15}$$

which is similar to the one presented by Bridgestone. Kerscher received in his experiments a good compliance between the model and real Festo muscles for high pressures. As used in the derivation of the Chou-Hannaford model, Kerscher also used a cylinder to approximate the volume of the muscle. To consider the fact that the muscle deformed at the end, he concluded that this must express in a different contraction. Therefore, he added a term to the contraction ratio:

$$F = \pi {r_0}^2 p \cdot (a(1-\epsilon(p)\kappa)^2 - b) \tag{2.16}$$

The following function has proven to be qualified:

$$\epsilon(p) = a_\epsilon \cdot e^{-p} - b_\epsilon \tag{2.17}$$

The parameters a_ϵ and b_ϵ have been chosen with the least squares method. If the muscle is expanded out of the initial position, which means that it has a negative contraction, it definitely would create a force because of the elasticity of the rubber. This can be caused by an initial tension when mounting the muscle in the joint. The formula calculates always $F = 0$ for $p = 0$, that is why he added a second part that is only active when the contraction ratio is negative. The second part can be described as

$$F_{add} = \sigma(-\kappa) \cdot (-f_0)\kappa \tag{2.18}$$

where σ is the unity step and f_0 is a constant. The following final equation was used in his thesis:

$$F = \pi {r_0}^2 p \cdot (a(1 - \epsilon(p)\kappa)^2 - b) + \sigma(-\kappa) \cdot (-f_0)\kappa \tag{2.19}$$

2.3.2 Control approaches for artificial muscles

Due to the complexity of artificial muscles, no control architecture is recommended. In fact, many different control strategies are possible, also depending on the application.

In [2], a regular PID controller was implemented to control pneumatic muscles. The performance of this controller was quite sensitive to errors caused by the muscle hysteresis or other errors in the feed-forward term. The same authors followed another approach in using an adaptive pole-placement controller [3]. A polynomial model whose parameters were estimated at each sampling interval was used to determine the controller's poles.

An approach to control artificial muscles with a Fuzzy Controller is developed in [4].

One problem in controlling artificial muscles is the development of an accurate muscle model. Therefore, it seems reasonable to apply a neural network to this problem. In [23], a feed-forward neural network was applied to one joint driven by pneumatic muscles. To judge the quality of the neural network control, van der Smagt used two *sin* trajectories to train the neural network. Van der Smagt showed that the neural network could follow the simple trajectory after only 16 training sessions with an average error of less than 0.1°.

These experiments showed that a neural network might be a good alternative to conventional control when using one joint and a periodic input function.

Other researchers implemented a Kohonen network to solve position control problems [12]. These approaches did not assert itself because they have been either too imprecise with a accuracy of only 1cm, or they needed too many training samples. For a satisfying teaching, at least 300 training samples have been necessary according to [21], which means several hours of training.

The attempt to realize the entire control of the arm with a neural network is questionable. One problem is that neural networks can only be trained for specific inputs. It is impossible within a limited time to teach a neural network all possible inputs of a six-degree-of-freedom arm to guarantee robustness and safety. The application of neural networks for controlling individual joints with reiterating control inputs might in contrast be useful.

Chapter 3

Actuator Modeling

3.1 Actuator testbed

To find the relationship between the muscle's system variables, to generate an actuator model and to verify it, a testbed was built (Figure 3.1).

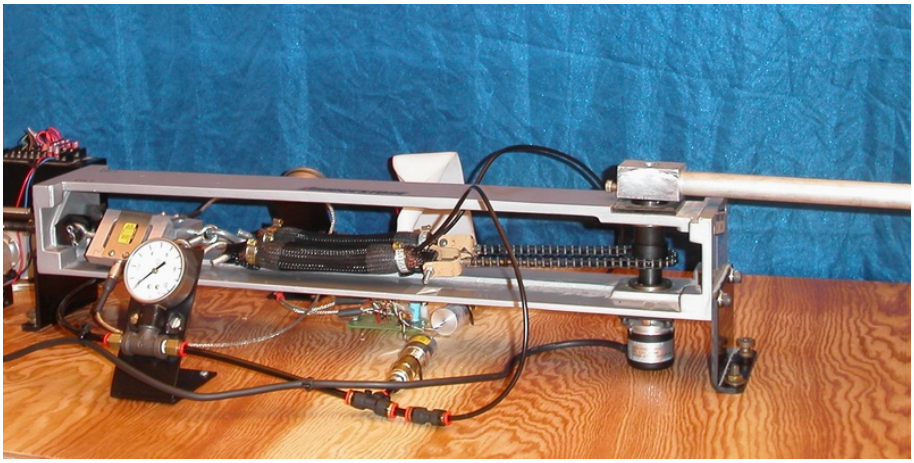

Figure 3.1: Pneumatic actuator testbed

To measure the resulting torque, two force sensors (Futek L2353 [8]) are connected between the end of the muscles and the testbed frame. External pressure sensors (Futek P4010, [8]) are placed as close to the muscles as possible to measure the real muscle pressures and to avoid time delays or pressure loss because of long tubes. An optical encoder (Sumtek, 8000 steps/turn) is used to determine the actual joint angle. Through an encoder buffer, the encoder signal is read by a PC interface board, which has been developed in the CIS lab [18]. Figure 3.1 illustrates an overview of the components.

Pressure is given from a Bridgestone servo valve unit (SVO 102-A06) that receives the desired pressures via a 4-20 mA signal from the PC interface card. A National Instruments multi-IO card (Lab PC 1200) reads the signal of force and pressure sensors into the PC.
The monitor output of the servo valve unit could have been used to gain information about the actual pressure in the tubes. But in this case, the distance between sensors and real muscles would have been long, and separate sensors directly at the muscles lead to better results.

Specifications and data sheets of the used components can be found in Appendix E.

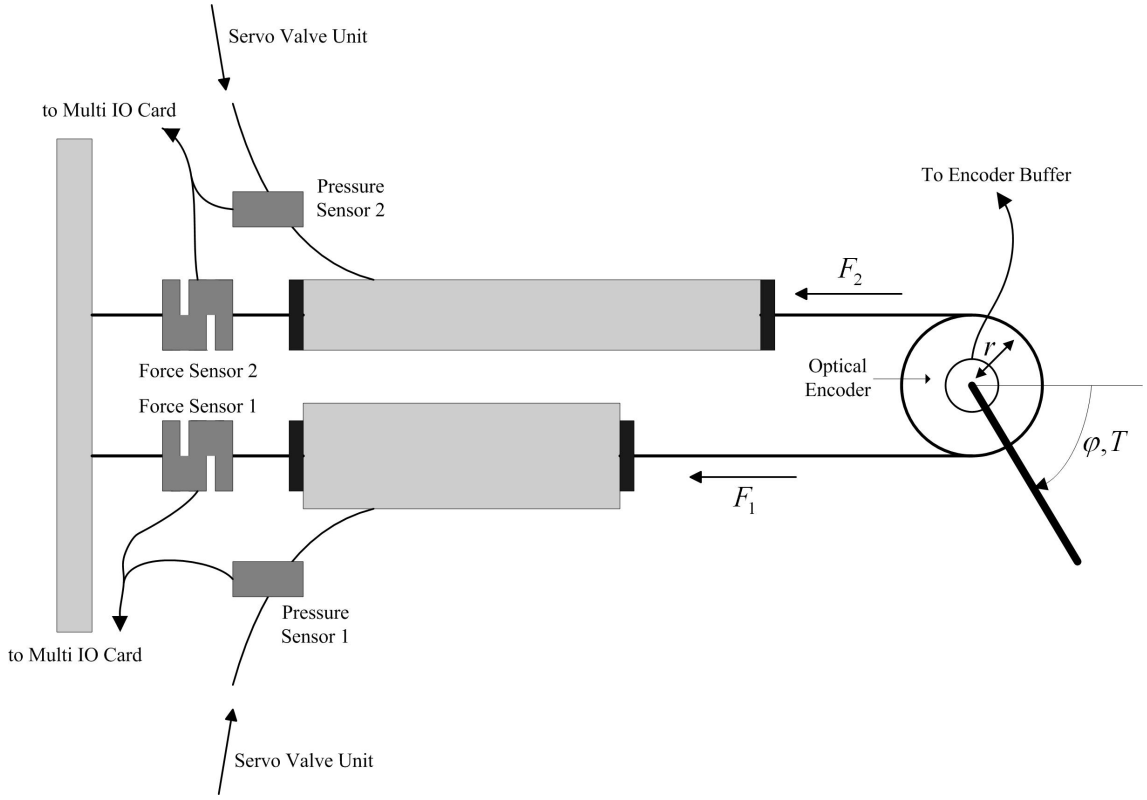

Figure 3.2: Testbed components overview

3.2 First experiments

As mentioned in Chapter 2, pneumatic actuators feature nonlinearity and hysteresis. To discuss the further procedure, the hysteresis is analyzed in some first experiments. Two issues have to be clarified by these experiments:

- Is it necessary to model the hysteresis itself?
- Is the hysteresis of the muscles velocity-dependent or not?

For the following experiments, one of the variables is kept constant while the second one isaltered and the third one is measured. Due to the fact that a constant torque can hardly be applied to the joint for different angles, there are two possible modes to run an experiment:

- Constant pressure mode, the pressure difference is set by the control unit, the torque is applied to the joint by hand; joint angle and torque are measured
- Constant angle mode, the arm of the joint is clamped, a varying pressure is applied to the muscles; the resulting torque is measured

Due to practical reasons, the experiments were mostly run in constant pressure mode, the constant angle mode has only been used later to verify the experiments. Both types have been run with different velocities in the following experiments.

We already assume that the pneumatic actuators contain, due to the friction between the inner rubber tube and the outer netting, a hysteresis. To find a convenient model, it is important to have information about the width and the type of the hysteresis. If the

hysteresis paths are not too far away from each other, the behavior might be modeled with the Bridgestone or other models introduced in Chapter 2.3.1. If the hysteresis is too wide, a way to model it must be found.

For this purpose, an experiment using a pair of Shadow 290 mm muscles with a constant pressure difference of 0 kg/cm^2 at a initial pressure of 3 kg/cm^2 was chosen. The muscles were mounted the way that they featured a pre-stressing of about 15 %. This has practical reasons to allow a roughly constant stiffness over the entire workspace because the maximum contraction ration is about 30%. In the first experiment, the arm was moved at a very slow speed about $0.1\frac{\pi}{s}$ from one end position to the other one several times. The result is shown in Figure 3.2.

Expecting a hysteresis for the quasi-static experiment, the data showed a nearly linear relationship between torque and joint angle. In this case, modeling of the hysteresis is not necessary, because the width itself is within measurement precision.

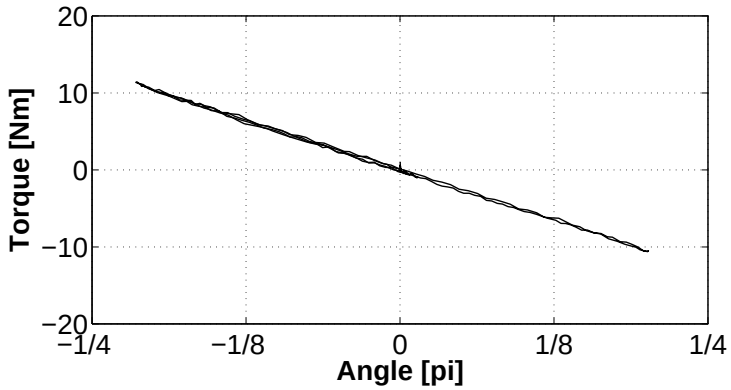

Figure 3.3: Torque-position relationship for slow speed

Most muscle models are physical static models. Only in [1], the model has been extended dynamically. Since rubber has also damping characteristics, the joint velocity might need to be considered in the actuator model. For that reason, the same experiment was now run with a speed of up to $2\frac{pi}{s}$. The result is a wider hysteresis, as shown in Figure 3.2.

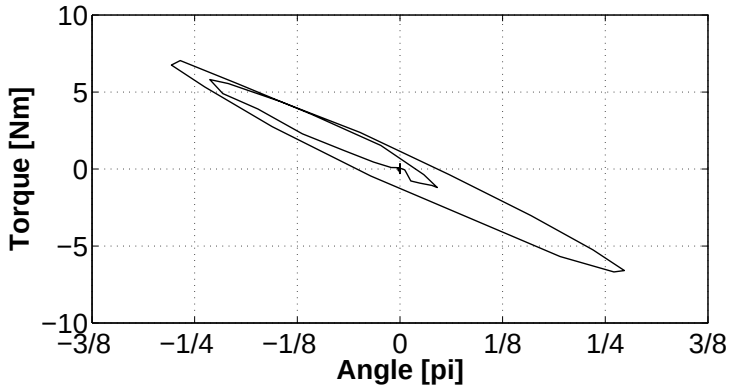

Figure 3.4: Torque-position relationship for high speed

At higher speed, a hysteresis appears that opens up to a difference of 2 Nm for the joint's middle position. The question is now where this hysteresis comes from. The assumption for

this experiment was, that the pressure in both of the muscles is constant. If the pressure varies with time, the hysteresis could be a consequence of that and not be generated by the muscles themselves. In reality, the progression of the pressure was not constant, as can be seen in Figure 3.2. Three peaks in the pressure diagram are caused by the fast movement from the middle position towards the end positions.

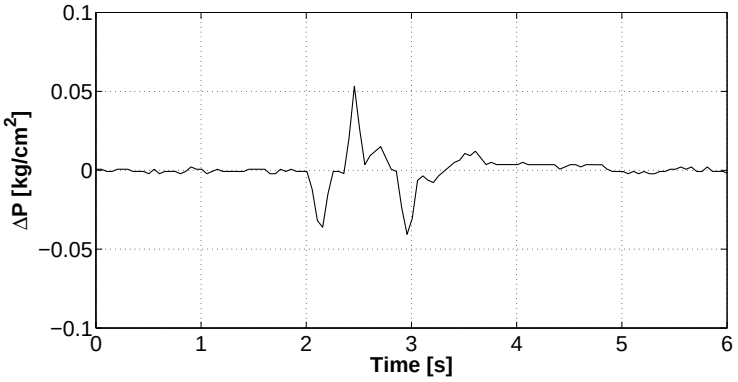

Figure 3.5: Pressure lapse for fast movement

To make sure that the hysteresis is not caused by a different pressure in the muscles, the pressure has to be kept constant. To avoid using the pressure control unit, which might be to slow for this experiment and therefore be the reason for the pressure peaks, the muscles have now been connected directly to the pressure supply via a wide tube which one acts as large capacitor.

The result of the fast movement was again a hysteresis, but the pressure was now constant. This result led to the conclusion that the hysteresis is not caused through the changing pressure because of a slow pressure control unit, but is a result of the damping characteristics of the rubber.

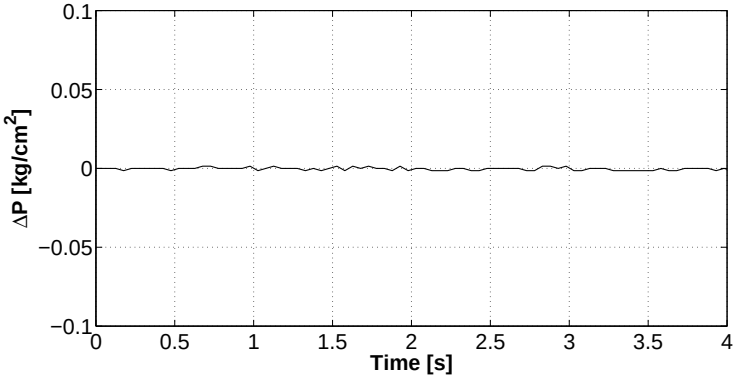

Figure 3.6: Pressure lapse after connecting muscles to a larger supply

These two experiments showed that the hysteresis can be disregarded for quasi-static states, but needs to be considered for fast movements. The static actuator model has to be extended with a dynamic part to make the model fit for a larger range of operation.

3.3 Modeling

3.3.1 Static actuator modeling

Most existing models are physical static models based on the approach of energy conservation [5, 13, 7]. All are very similar and can partly be transformed from one into another. One possibility to describe the relationship between force F, pressure p and contraction ratio ϵ of one muscle is the Bridgestone model, as described in Section 2.3.1. This model was used as a first basis for a static actuator model:

$$F = {D_0}^2 p[a(1-\epsilon)^2 - b] \tag{3.1}$$

Where F is the force generated by the muscle, p is the internal muscle pressure, and ϵ is the contraction ratio. D_0 is the muscle diameter before displacement, a and b are parameters. Combining the two muscles and using the relations 2.1.1, as well as

$$\epsilon = \frac{l_{max} - l}{l_{max}} \tag{3.2}$$

and

$$l_{1,2} = l_0 \pm r\varphi, \tag{3.3}$$

the new formula for the joint is:

$$\begin{aligned} T = \frac{2{D_0}^2 a r^3}{{l_0}^2}\Delta p\varphi^2 - \frac{2{D_0}^2(p_1+p_2)ar^2}{l_0}\varphi \\ + {D_0}^2 r(2a-2b)\Delta p \end{aligned} \tag{3.4}$$

or the simplified form by transferring D_0^2, l_0 and $p_1 + p_2 = 2p_0$ into the parameters a and b:

$$T = a'\Delta p r^3\varphi^2 + b' r^2\varphi + c'\Delta p r \tag{3.5}$$

Where a', b' and c' are new joint parameters, r is the radius of the chain wheel, $l_{1,2}$ are the muscle lengths and l_0 is the initial muscle length. The radius r of the chain wheel is different in testbed and real joints, it can therefore not be identified within the parameters a', b' and c'.

In the next experiment, different Δp's have been applied to the muscles in the testbed. The bar which is attached to the joint was moved very slowly by hand to record the quasi-static relation between torque and joint angle for each value of Δp. The result in Figure 3.7 showed a nearly linear relationship for the entire range of Δp, the lines have only been shifted in horizontal direction.
The first part in equation 3.5 would have only an appreciable effect on the torque for larger joint angles and higher Δp's. According to the experimental results in Figure 3.7, this part does not deserve any interest and does not affect the behavior of the actuator significantly. Equation 3.5 is therefore changed to:

$$T = b' r^2\varphi + c'\Delta p r \tag{3.6}$$

or, written with new parameters

$$T = Ar^2\varphi + B\Delta p r \tag{3.7}$$

Parameters A and B have been identified as A=-23450 $\frac{N}{m}$ and B=174.2 $N\frac{cm^2}{kg}$.

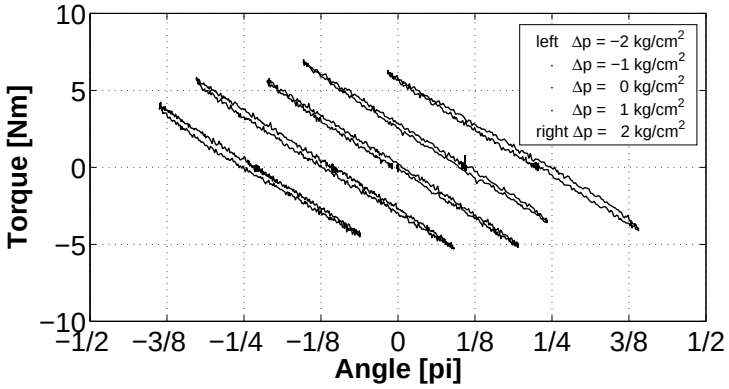

Figure 3.7: Quasi-static behavior for different Δp's

3.3.2 Dynamic actuator modeling

The results of the fast movement experiment in Figure 3.2 show a distance between the two paths of up to 1 Nm for the highest velocity. Under these circumstances, the hysteresis cannot be approximated with a single line and has to be modeled itself.

Because the arm is moved manually and restricted to the end positions, the velocity of the joint changes during the entire movement and cannot be kept constant. This allows no conclusion about the form of the hysteresis curve and the relation to the driven velocity. To gain further information, the arm was moved for a longer period of time, approximately 5 minutes, in all positions and with different speeds. Result of this experiment was a completely filled hysteresis curve containing lots of different velocities and positions (Figure 3.8).

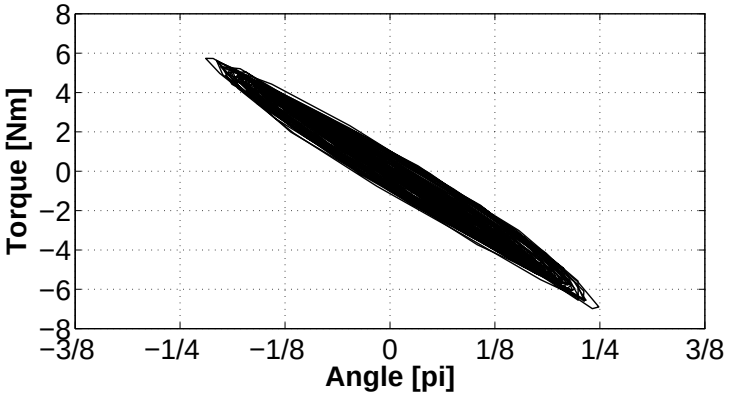

Figure 3.8: Bundle of hysteresis curves containing all different velocities

Out of this bundle, only points with constant velocities were extracted and plotted in the angle-torque diagram. Figure 3.9 shows the results for two different velocities, including both directions.
The points with constant velocities lay on straight lines with the same gradient than the static main line once again. The velocity-dependence is finally only an offset to the static model. Equation 3.7 is therefore extended to:

$$T = Ar^2\varphi + Br\Delta p + f(\dot{\varphi}) \tag{3.8}$$

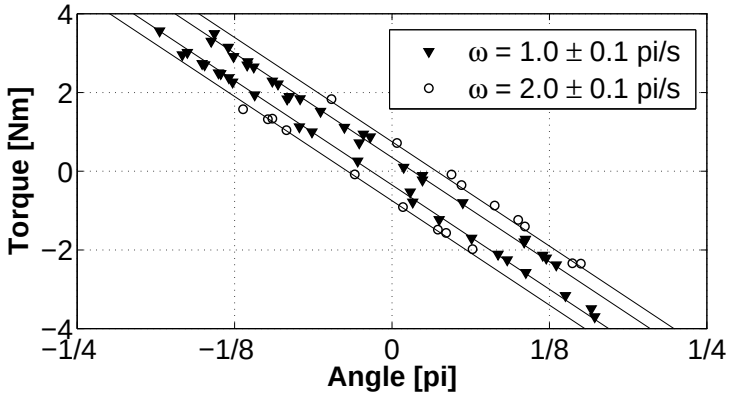

Figure 3.9: Constant velocities on straight lines

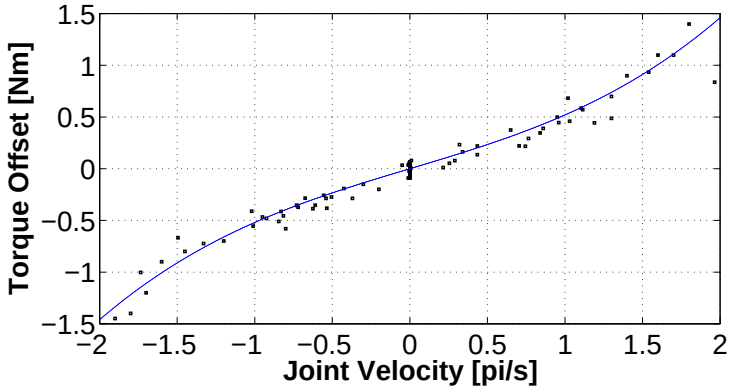

Figure 3.10: Torque offset over joint velocity

To find out the relationship between this offset and the corresponding velocity, the bundle of hysteresis was cut at $\varphi = 0\pi$ and the section plotted in Figure 3.10. The middle position of the joint was chosen because this point contains most of the appearing velocities. To make sure that the allocation did not change over the joint position, the results were verified with cuts at other positions. The following function is qualified for interpolating the data in Figure 3.10:

$$f(\dot{\varphi}) = C\dot{\varphi}^3 + D\dot{\varphi} \tag{3.9}$$

The parameters of this function were identified as C=0.0026 Nms^3 and D=0.14 Nms.

Finally, the complete equation representing the actuator model is:

$$T = Ar^2\varphi + Br\Delta p + C\dot{\varphi}^3 + D\dot{\varphi} \tag{3.10}$$

3.4 Model verification

To verify the actuator model in equation 3.10, two different experiments were performed. First, the joint bar was moved by hand so that the model was acting as a sensor and had to

predict the applied torque. At the same time, the force sensors measured the real torque. The pressure of the muscles was adjusted to $\Delta p = 0\ \frac{kg}{cm^2}$. The result is illustrated in Figure 3.11.

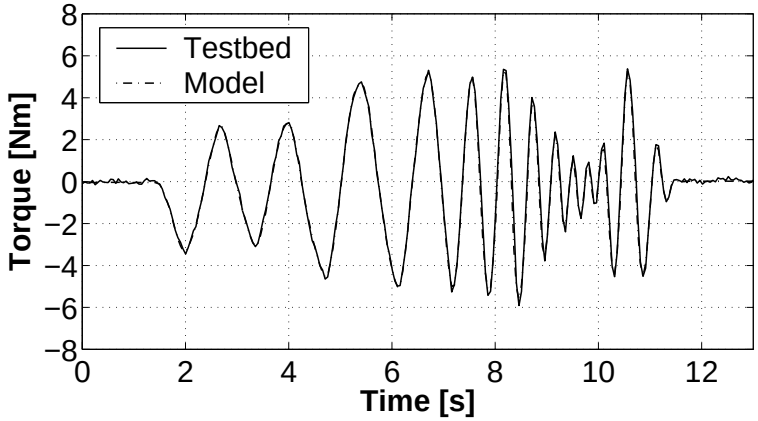

Figure 3.11: Measured torque compared with prediction of actuator model

The first experiment was repeated with different Δp's to guarantee the functionality for extreme joint positions. As long as the arm was not moved until one of the artificial muscles lost the pre-tension completely, the actuator model prediction and the real measurement results resembled each other.

In the second type of experiment, the joint bar was fixed in one position. A desired torque was given to the actuator model, which one had to calculate the needed Δp to apply this torque. The desired value was applied as a step input and the answer was recorded by the force sensors. Figure 3.12 shows the results for a fixed middle position of the joint and different input values.

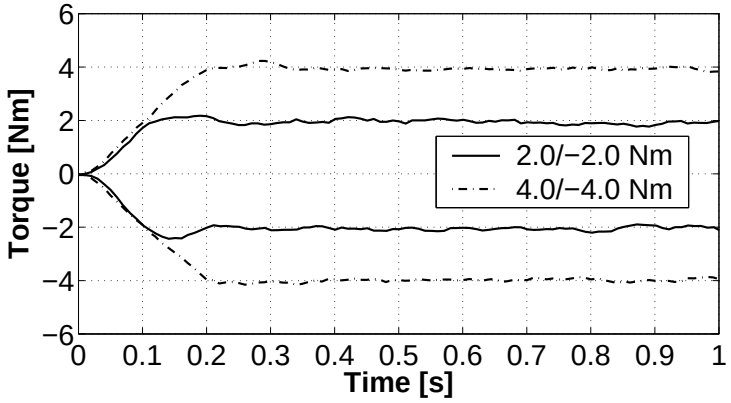

Figure 3.12: Various step inputs for torque controller

Due to time delays of the servo valves and the compressible air, it took about 0.15 s to apply a torque of 2 Nm and 0.25 s for a desired torque of 4 Nm. These time delays are very low for a pneumatic controlled system and may increase on ISAC where longer tubes are needed. The desired torque was reached with an accuracy of about 0.1 Nm.

To judge the influence of the dynamic part of equation 3.10, the experiment in Figure 3.11 was repeated and the prediction of the actuator model was generated with and without the dynamic part. Initially, the results do not differ very much from the ones in Figure 3.11.

However, after a closer look at the areas with high velocity, the model without dynamic part predicts a higher torque if a positive torque is applied, (means that the arm was moved in positive direction), and a lower torque on the decreasing path of the hysteresis (Figure 3.13). The error between static and dynamic behavior is caused exactly by the width of the hysteresis, so the full actuator model almost totally compensates this error.

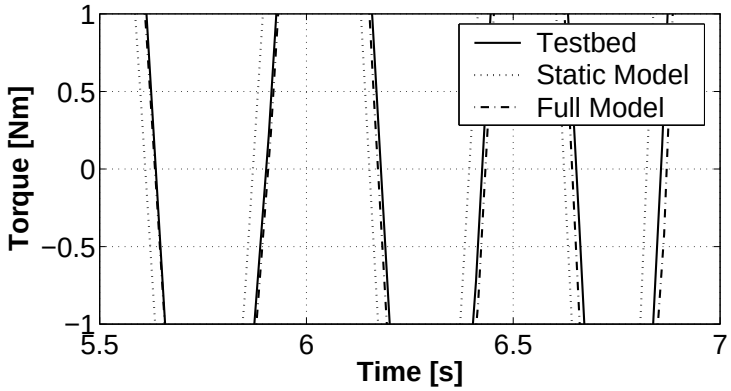

Figure 3.13: Comparison of static model and complete model including static and dynamic parts

Experimental results showed that the actuator model predicts the torque very well for different velocities, joint positions and muscle pressures. Because the feedback of the torque is missing, it is very important to carefully identify the model parameters and to adjust the right pre-tension when mounting the muscles in the arm. Otherwise the parameters will change and cause incorrect input data for the torque controller.

Another issue is that the pressure cannot be measured at the muscles in ISAC. The real pressure may differ from the desired pressure in ISAC because of longer air tubes used between the pressure control unit and the muscles in the testbed. In this case, the pressure control unit and the tubes would have to be modeled to consider their time delay.

The velocity of the joint depends mainly on the pressure control unit. With the Bridgestone servo valves, it is hardly possible to reach a joint velocity of more than $1\frac{pi}{s}$. However, it is important to have an accurate model that fits a large range of operation.

3.5 Restrictions

The actuator model was derived by using Shadow muscles. Experiments showed the applicability for Bridgestone Rubbertuators, other brands have not been tested. The model is valid in the range of $\Delta p = \pm 2\ \frac{kg}{cm^2}$. It is no more accurate for a larger range of Δp, or a joint angle larger than $\pm\frac{3}{8}\pi$. If the joint angle is too large, one of the muscles looses its pre-tension completely which leads to a deviation of the straight line in Figure 3.7. Note that the restrictions of joint angle limitations change with another diameter of the chain wheel (on ISAC).

All information refers to an initial pressure of 3.0 $\frac{kg}{cm^2}$, other initial pressure will result in either earlier tension-loss and therefore restriction of Δp or restrictions in joint angle operation range. The pre-tension when mounting the muscles has to be within the range of 15 ± 3 % to guarantee proper function of the model.

Chapter 4

System Modeling

In this section, modeling of arm and system parts is described. Transfer functions of the different system parts are required to design the controller and to proof stability. The system can be divided into pressure control unit, artificial muscles (actuator) and the joint itself. As shown in Figure 4.1, transfer functions of actuator and arm are coupled. A kinematic model of the arm is needed to solve direct and inverse kinematic problems.

4.1 Pressure control unit

The transfer function of pressure control unit (PCU) was determined experimentally. The air tubes are assumed to be part of the PCU to simplify the system identification. A reference current value, corresponding to 5 $\frac{kg}{cm^2}$ was given as a step input to the pressure control unit. Figure 4.2 shows the response, which identified the pressure control unit as a first order lag element.
The equation for the PCU transfer function is therefore of the form:

$$F_C(s) = \frac{K_C}{1 + T_C s} \tag{4.1}$$

The two parameters K_C and T_C were identified as $K_C = 1$ and $T_C = 0.17$ s, so that the final transfer function is:

$$F_C(s) = \frac{1}{1 + 0.17s} \tag{4.2}$$

The experiments were carried out with a tube length of about 5 m between pressure control unit and artificial muscles. This length matches approximately the length of the air tubes in ISAC. The lag time T_C is, depending on the reference pressure or a changing tube length,

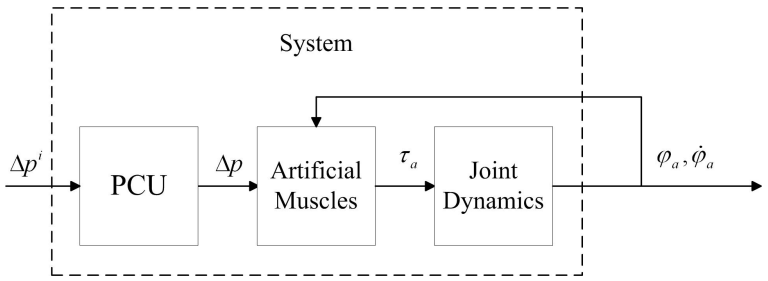

Figure 4.1: System parts

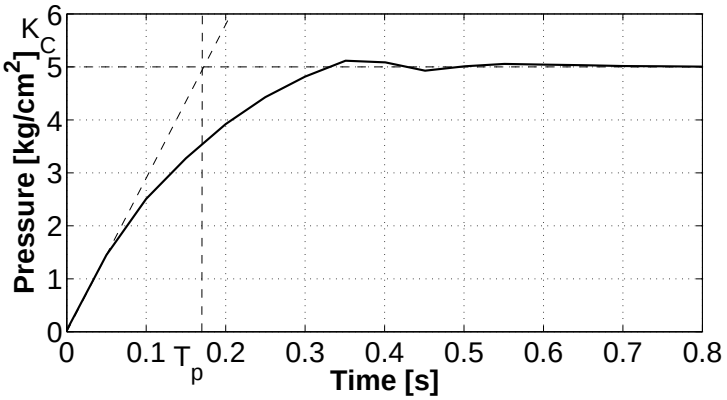

Figure 4.2: Step input response of the pressure control unit

sometimes shorter than 0.17 s, so that this is representing the worst case. Since we deal with the actuator element, the real time lag is allowed to be shorter as assumed because the control speed and quality would benefit if so.

4.2 Joint modeling

The simple equation of motion for the joint is:

$$I\ddot{\varphi} - \tau = 0 \tag{4.3}$$

or with Equation 3.10 and disregard of $C\dot{\varphi}^3$ what affects only high velocities:

$$-\frac{I}{Ar^2}\ddot{\varphi} - sign(\dot{\varphi})\frac{D}{Ar^2}\dot{\varphi} + \varphi = -\frac{B}{Ar}\Delta p \tag{4.4}$$

Where I is the moment of inertia of the joint which is mainly given through a metal bar, used to indicate the joint position. The moment of inertia was computed as I=0.0085 kgm^2, referred to the joint axis. Equation 4.4 has the structure of a second order lag time element. A comparison of coefficients with a regular form of a second order element

$${T_J}^2\ddot{\varphi} + 2\zeta T_J\dot{\varphi} + \varphi = K_J\Delta p \tag{4.5}$$

leads to the following values:

$T_J = \sqrt{-\frac{I}{Ar^2}} = 0.032s$, $\zeta = \frac{D}{2Ar^2\sqrt{-\frac{I}{Ar^2}}} = 0.26$ and $K_J = -\frac{B}{Ar} = 0.3$.

The damping coefficient ζ as well as the time constant T_J have to be positive, which means that $sign(\dot{\varphi})$ is assumed always positive. This makes also sense in the way that the damping property is working against the system movement, not depending on the direction.

The resulting transfer function of the joint is therefore:

$$F_J(s) = \frac{K_J}{1 + 2\zeta T_J{\cdot}s + {T_J}^2{\cdot}s^2} \tag{4.6}$$

or with inserted parameters:

$$F_J(s) = \frac{0.3}{1 + 0.018 \cdot s + 0.001 \cdot s^2} \tag{4.7}$$

To judge the controllability of a system and to tune control parameters, the relation of the parameters latency T_L and adjustment time T_A are helpful [16]. The latency is the time that passes, until the changing input value causes an output, which is appreciable different from zero. This point is defined as the intersection of the time axis and the tangent in the inflection point of the system response. The adjustment time is the time that passes between the dead time until the tangent at the inflection point cuts the abscissa given through K_J.

To determine these two variables, the step response of the joint was recorded. A problem when accomplishing this experimentally is, that it is de facto not possible to give a step to the actuator because the pressure control unit has a first-order delay element behavior as shown in Section 4.1. Another possibility to give a step input to the muscles might be switching directly to an air supply with the reference pressure and a sufficient capacity. The input function would probably have a shorter delay time, but still be of a first-order delay time behavior due to air compressibility and friction in the wires.

One way to see the response is simulating a step to the existing transfer function. With Matlab/Simulink, a step input of Δp=2 $\frac{kg}{cm^2}$ was given as input to the transfer function 4.6 of the joint. The system response is shown in Figure 4.3.

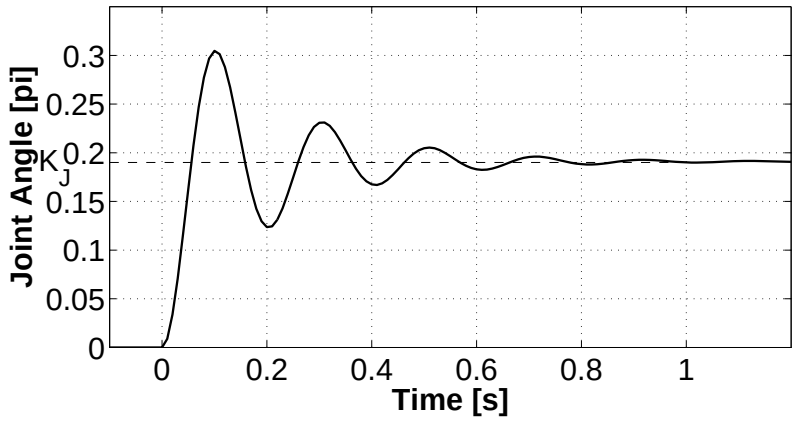

Figure 4.3: Simulated step response of joint transfer function

Figure 4.4 shows the parameters latency and adjustment time, which have been identified as $T_L = 0.015$ s and $T_A = 0.045$ s.

According to [16], the controllability of a system is getting more and more difficult, the higher T_L is, because the controller cannot interact and the system is totally committed to an occurring disturbance, and the lower T_A is, what means that the system responds fast after the latency is over and the controller has to act also fast and adequate.

The relation T_L/T_A is an indicator for the controllability of a system. In our case, a relation of $T_L/T_A = 0.36$ is controllable, but needs a high control effort [16]. This could be an indicator that we have to fall back on higher control architectures. Easy controllable systems have a T_L/T_A ratio of 0-0.1, these with over 0.8 are hardly controllable. The parameters T_L and T_A can further be used to tune control parameters.

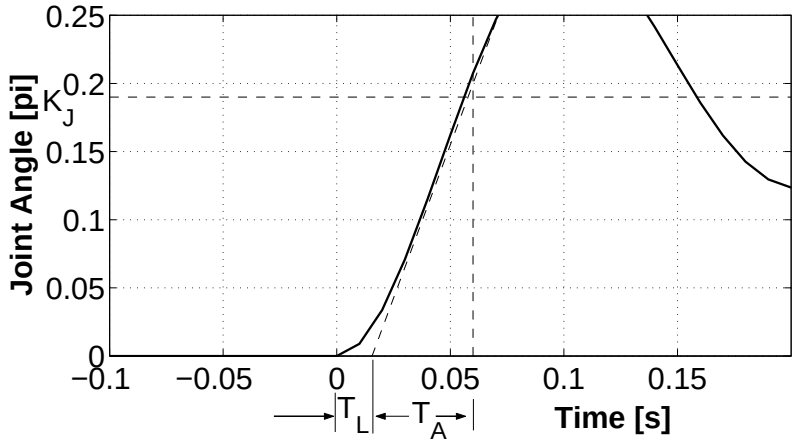

Figure 4.4: Closer view on simulated response

4.3 Arm Kinematics

The robot arm has a serial kinematic, similar to lots of industrial robots. The relationship between different links can be described with a 4 x 4 homogeneous transformation matrix, containing a 3 x 3 rotation matrix R, an 1 x 3 vector R pointing from frame base $i-1$ to frame base i. To obtain a square matrix, the last row is filled with [0 0 0 1]. Changing these values allows perspective or scaling operations but this is not of interest for this work.

$$T = \left[\begin{array}{c|c} R & P \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \tag{4.8}$$

The Denavit-Hartenberg notation is one possibility to describe the relationship between different joint links, and to obtain the homogeneous transformation matrices. Using this convention has the advantage that a transformation from one link to another, including rotation and translation of the basis, can be described with only four parameters instead of six.
Since this is a very common method, no details are included in this work. See [10] for more information about Denavit-Hartenberg notation and attaching the link frames correctly. Figure 4.3 shows the robot manipulator and the link frames, placed according to this convention.

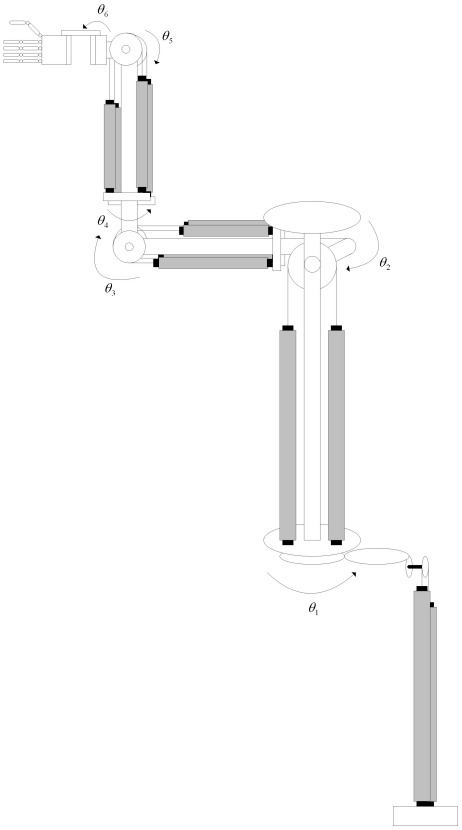

Figure 4.5: Soft Arm schematics

IMPORTANT NOTE: Figure 4.3 represents the right Soft Arm. To be able to use direct and inverse kinematics for the left arm, only the parameter $d_3 = l_2$ has to be changed to $d_3 = -l_2$. This leads to other numerical values in transformation matrices and solutions for direct and

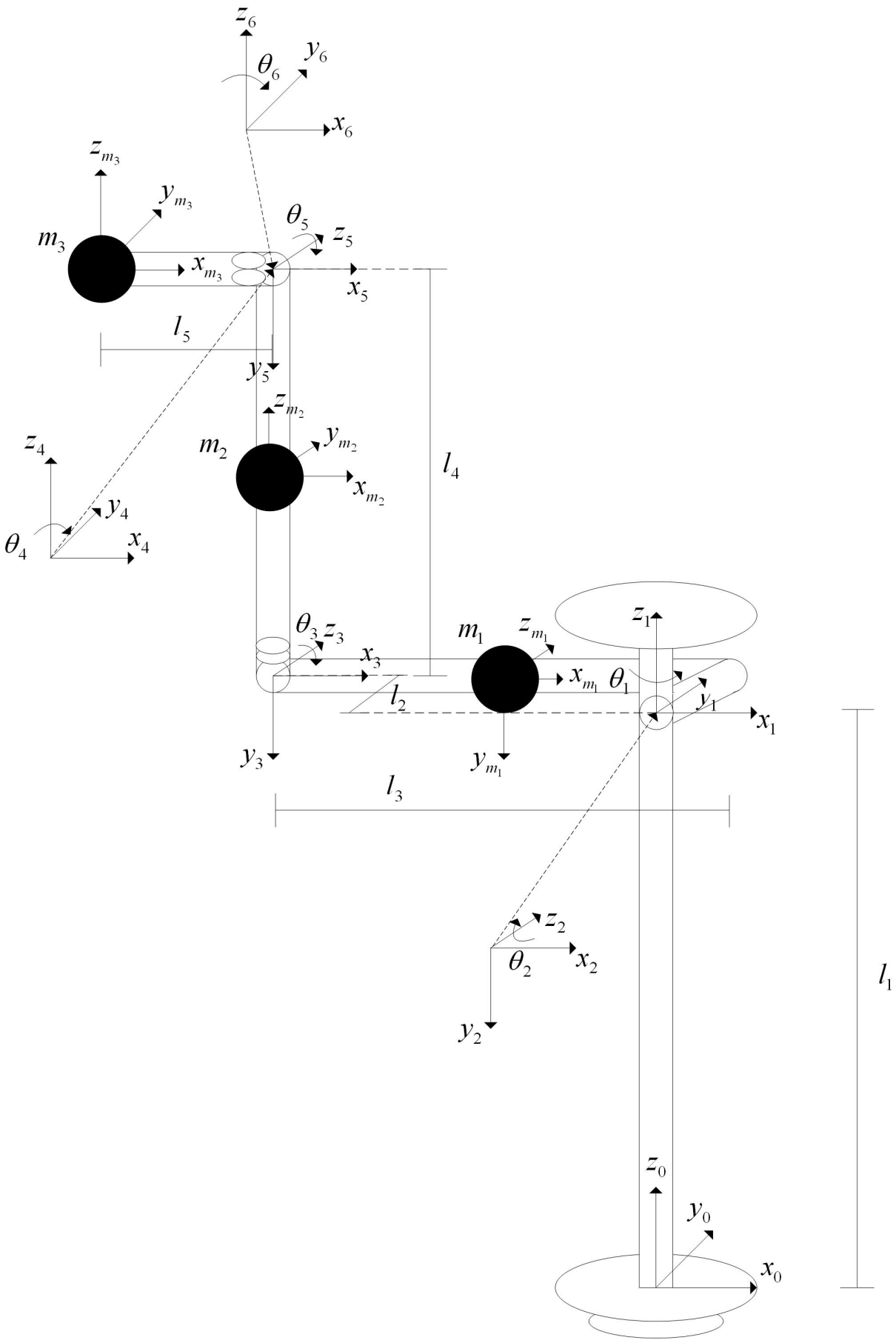

Figure 4.6: Link frames set according to Denavit-Hartenberg convention

l_1	=	0.510 m	m_1	=	1.750 kg
l_2	=	0.200 m	m_2	=	0.450 kg
l_3	=	0.330 m	m_3	=	0.140 kg
l_4	=	0.290 m			
l_5	=	0.160 m			

Table 4.1: Soft Arm dimensions and masses

inverse kinematics, but the same equations can be used.
Three masses, used as balance weights representing the self-weight of the arm are attached between joint two and joint three, joint three and joint four and in the tool center point. The position for mass m_3 was chosen as tool center point because it is easy to consider the weight of a load by simple addition to the mass m_3. The weight of this load can be determined with a torque sensor placed in the wrist part.
After attaching the link frames, the parameters have to be assigned as following [10]:

a_i	=	distance from Z_i to Z_{i+1} measured along X_i
α_i	=	angle between Z_i and Z_{i+1} measured about X_i
d_i	=	distance from X_{i-1} to X_i measured along Z_i
Θ_i	=	angle between X_{i-1} and X_i measured about Z_i

Table 4.2: Definition of link parameters

The link parameters can now be derived from Figure 4.3 as:

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	0	0	l_1	Θ_1
2	$-90°$	0	0	Θ_2
3	0	$-l_3$	l_2	Θ_3
4	$90°$	0	$-l_4$	Θ_4
5	$-90°$	0	0	Θ_5
6	$90°$	0	0	Θ_6
m_1	0	-0.5 l_3	l_2	0
m_2	$90°$	0	-0.5 l_4	0
m_3	0	-l_5	0	0

Table 4.3: Transformation parameters

The parameters of bases frames m_1 to m_3 are transformed from the previous links (1,3,6). Inserting the parameters in mask 4.9 leads to the transformation matrices from one joint to the next one. (see Appendix G for all matrices ${}^0_1T$ to ${}^6_{m_3}T$). All other matrices (e.g. ${}^0_3T$) can be computed by simple multiplication of the individual ones.

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos(\Theta_i) & -sin(\Theta_i) & 0 & a_{i-1} \\ sin(\Theta_i)cos(\alpha_{i-1}) & cos(\Theta_i)cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & -sin(\alpha_{i-1})d_i \\ sin(\Theta_i)sin(\alpha_{i-1}) & cos(\Theta_i)sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9} $$

To simplify the next equations, the following substitutions are used:

$$\begin{aligned} s_1 &= sin(\Theta_1) \\ s_2 &= sin(\Theta_2) \\ s_3 &= sin(\Theta_3) \\ c_1 &= cos(\Theta_1) \\ c_2 &= cos(\Theta_2) \\ c_3 &= cos(\Theta_3) \\ c_{23} &= cos(\Theta_2)cos(\Theta_3) - sin(\Theta_2)sin(\Theta_3) \\ s_{23} &= cos(\Theta_2)sin(\Theta_3) + sin(\Theta_2)cos(\Theta_3) \end{aligned} \tag{4.10}$$

All formulas used in following Sections 4.3.1 and 4.3.2 can be reviewed in the Maple file *'kinematics.mws'*.

4.3.1 Direct kinematics

The direct kinematics compute a XYZ position according to the actual joint angles. Usually, the xyz coordinates of the Tool Center Point (TCP), or the base of the last link frame are chosen.

The transformation matrix from link zero to mass m_3 (which has been placed in the TCP) has to be computed to derive the belonging XYZ position. Multiplying the individual transformation matrices according to

$$ {}^{0}_{m_3}T = {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T\ {}^{6}_{m_3}T \tag{4.11}$$

leads to the XYZ coordinates for the TCP:

$$\begin{aligned} x_{tcp} = {}^{0}_{m_3}T[1,4] &= -(((c_1c_{23}c_4 - s_1s_4)c_5 - c_1s_{23}s_5)c_6 + \\ &\quad (-c_1c_{23}s_4 - s_1c_4)s_6)l_5 - (-c_1s_{23})l_4 - c_1c_2l_3 - s_1l_2 \\ y_{tcp} = {}^{0}_{m_3}T[2,4] &= -(((s_1c_{23}c_4 + c_1s_4)c_5 - s_1s_{23}s_5)c_6 + \\ &\quad (-s_1c_{23}s_4 + c_1c_4)s_6)l_5 + s_1s_{23}l_4 - s_1c_2l_3 + c_1l_2 \\ z_{tcp} = {}^{0}_{m_3}T[3,4] &= (-(s_{23}c_4c_5 + s_{23}s_5)c_6 + \\ &\quad s_{23}s_4s_6)l_5 + c_{23}l_4 + s_2l_3 + l_1 \end{aligned} \tag{4.12}$$

The XYZ coordinates of frame six origin are computed in the same way, except the last multiplication

$$ {}^{0}_{6}T = {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T \tag{4.13}$$

and lead to the following XYZ coordinates for frame base six origin:

$$\begin{aligned} x_{base6} &= {}^{0}_{6}T[1,4] = c_1s_{23}l_4 - c_1c_2l_3 - s_1l_2 \\ y_{base6} &= {}^{0}_{6}T[2,4] = s_1s_{23}l_4 - s_1c_2l_3 + c_1l_2 \\ z_{base6} &= {}^{0}_{6}T[3,4] = c_{23}l_4 + s_2l_3 + l_1 \end{aligned} \tag{4.14}$$

4.3.2 Inverse kinematics

The inverse kinematics are used to compute the belonging joint angles to a desired XYZ position. Solving the inverse kinematics is not a simple problem. In our robot ISAC, the six degrees of freedom give us twelve equations in the ${}^{0}_{6}T$ matrix, three from the position vector and nine from the 3x3 rotation matrix. Because there are only six unknowns, multiple solutions will result when solving the equations.

The ISAC kinematics are close to the PUMA 560 kinematics, which ones have been solved in [10]. The transformations are different, but the approach is the same. First, all existing solutions are computed, later the number of solutions (if there are any) is reduced by considering joint limitations and finally a solution is picked by computing the shortest distance to the actual arm position.
The input for our equations are numeric values given by the transformation matrix

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.15} $$

To find a solution for the position of the first joint, Equation 4.13 is written as

$$ [{}^{0}_{1}T]^{-1}\ {}^{0}_{6}T = {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T \tag{4.16} $$

or

$$ \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{1}_{6}T \tag{4.17} $$

Elements $[2,4]$ from both sides of Equation 4.17 can be written as:

$$ -s_1 p_x + c_1 p_y = l_2 \tag{4.18} $$

Using the same trigonometric substitutions as in [10] leads to two solutions for Θ_1:

$$ \Theta_1 = Atan2(p_y, p_x) - Atan2(l_2, \pm\sqrt{p_x^2 + p_y^2 + p_z^2}) \tag{4.19} $$

Equating elements $[1,4]$ and elements $[3,4]$ in 4.17 results in the equations

$$ \begin{aligned} c_1 p_x + s_1 p_y &= s_{23} l_4 - c_2 l_3 \\ p_z - l_1 &= c_{23} l_4 + s_2 l_3 \end{aligned} \tag{4.20} $$

Squaring Equations 4.18 and 4.20 and adding the results leads to

$$ p_x^2 + p_y^2 + p_z^2 - 2 p_z l_1 + l_1^2 = l_2^2 + l_4^2 - 2 l_4 l_3 s_3 + l_3^2 \tag{4.21} $$

or the final solutions for Θ_3:

$$\begin{aligned}\Theta_{3,1} &= asin(\frac{-p_x^2 - p_y^2 - p_z^2 + 2p_z l_1 - l_1^2 + l_2^2 + l_3^2 + l_4^2}{2l_3 l_4}) \\ \Theta_{3,2} &= pi - asin(\frac{-p_x^2 - p_y^2 - p_z^2 + 2p_z l_1 - l_1^2 + l_2^2 + l_3^2 + l_4^2}{2l_3 l_4})\end{aligned} \tag{4.22}$$

Note: Because the asin function returns only a value between $-\frac{pi}{2}$ and $\frac{pi}{2}$, the second solution has to be computed separately. Two different solutions for Θ_1 lead to four solutions for Θ_3. After Θ_1 and Θ_3 are known, Equation 4.13 can be written as

$$[{}^0_3T]^{-1}\ {}^0_6T = {}^3_4T\ {}^4_5T\ {}^5_6T \tag{4.23}$$

or

$$\begin{bmatrix} c_{23}c_1 & c_{23}s_1 & -s_{23} & c_3l_3 + s_{23}l_1 \\ -s_{23}c_1 & -s_{23}c_1 & -c_{23} & -s_3l_3 - c_{23}l_1 \\ -s_1 & c_1 & 0 & -l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^3_6T \tag{4.24}$$

Equating $[1,4]$ and $[2,4]$ elements gives

$$\begin{aligned} c_1c_{23}p_x + s_1c_{23}p_y - s_{23}p_z + c_3l_3 + s_{23}l_1 &= 0 \\ -c_1s_{23}p_x - s_1s_{23}p_y - c_{23}p_z - s_3l_3 + c_{23}l_1 &= -l_4 \end{aligned} \tag{4.25}$$

These two equations can now be solved for s_{23} and c_{23}, which leads to the very unattractive equations

$$c_{23} = \frac{(-c_1c_3p_x - s_1c_3p_y - s_3p_z + s_3l_1)l_3 + (p_z - l_1)l_4}{c_1^2p_x^2 + s_1^2p_y^2 + p_z^2 + 2c_1s_1p_xp_y - 2l_1p_z + l_1^2} \tag{4.26}$$

and

$$s_{23} = -\frac{(c_1s_3p_x + s_1s_3p_y - c_3p_z + c_3l_1)l_3 + (-c_1p_x - -s_1p_y)l_4}{c_1^2p_x^2 + s_1^2p_y^2 + p_z^2 + 2c_1s_1p_xp_y - 2l_1p_z + l_1^2} \tag{4.27}$$

According to the four possible solutions for Θ_{23}, the solutions for Θ_2 can be computed as

$$\Theta_2 = \Theta_{23} - \Theta_3 \tag{4.28}$$

The computation of joint angles Θ_4, Θ_5 and Θ_6 is similar to the Puma inverse kinematics. According to [10], the solution for these angles is:

$$\Theta_4 = Atan2(-r_{13}s_1 + r_{23}c_1, -r_{13}c_1c_{23} - r_{23}s_1c_{23} + r_{33}s_{23}) \tag{4.29}$$

$$\Theta_5 = Atan2(s_5, c_5) \tag{4.30}$$

whereas

$$\begin{aligned} s_5 &= -r_{13}(c_1c_{23}c_4 + s_1s_4) - r_{23}(s_1c_{23}c_4 - c_1s_4) + r_{33}(s_{23}c_4) \\ c_5 &= r_{13}(-c_1s_{23}) + r_{23}(-s_1s_{23}) + r_{33}(-c_{23}) \end{aligned} \tag{4.31}$$

and finally

$$\Theta_6 = Atan2(s_6, c_6) \tag{4.32}$$

whereas

$$\begin{aligned} s_6 &= -r_{11}(c_1c_{23}s_4) - r_{21}(s_1c_{23}s_4 + c_1c_4) + r_{31}(s_{23}s_4) \\ c_6 &= r_{11}[(c_1c_{23}c_4)c_5 - c_1s_{23}s_5] + r_{21}[(s_1c_{23}c_4 - c_1s_4)c_5 - s_1s_{23}s_5] - \\ &\quad r_{31}(s_{23}c_4c_5 + c_{23}s_5) \end{aligned} \tag{4.33}$$

Considering the fact that the wrist can be flipped, four more solutions for each of the angles Θ_4, Θ_5 and Θ_6 are obtained.

$$\begin{aligned} \Theta_4^* &= \Theta_4 + 180^\circ \\ \Theta_5^* &= -\Theta_5 \\ \Theta_6^* &= \Theta_6 + 180^\circ \end{aligned} \tag{4.34}$$

After computing all eight solutions, joint limitations are used to eliminate some of them. Out of the remaining ones, the closest one to the actual joint position is chosen to be the solution of the inverse kinematics.

4.4 Compensation of static and dynamic effects

Building a dynamic model for the entire arm takes a lot of time and is probably not worth the effort. Most of control architectures for industrial robots are designed as linear controllers without complete arm model. The trend is to make the control faster rather than trying to describe the system exactly. However, it is possible to consider some static and dynamic effects to improve the control without taking a lot of effort. All formulas used in this section can be reviewed in the Maple file *'statdyneffects.mws'*.

Forces, or finally torques, acting as large disturbances to the control, are:

- Torque caused by moment of inertia
- Torque caused by gravity

In our case, the torque caused by the moment of inertia is large for joint number one, because all masses are turning around this joint. Large gravity effects are acting on joint number two and joint number three. Influence of gravity on joint numbers four to six may not be grave, but are included for the sake of completeness.

Implementation and compensation of torques caused by the masses m_1 to m_3 in the control are subject of Section 5.4 in Chapter 5. In this section, only computations of these torques are prepared.

4.4.1 Influence of masses on moment of inertia for first joint

To compute the mass of inertia for the first joint, the distance of the mass point to the axis of joint one is needed. This is done by multiplying the transformation matrices from joint one to the corresponding mass, and computing the distance out of the X and Y coordinates stored in matrices ${}^{1}_{m_i}T[1,4]$ and ${}^{1}_{m_i}T[2,4]$.

After building the transformation matrices from joint one to masses m_1 to m_3 (see Appendix G) according to

$$\begin{aligned} {}^{1}_{m_1}T &= {}^{1}_{2}T\ {}^{2}_{m_1}T \\ {}^{1}_{m_2}T &= {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{m_2}T \\ {}^{1}_{m_3}T &= {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T\ {}^{6}_{m_3}T \end{aligned} \tag{4.35}$$

the distances to joint axis one are:

$$\begin{aligned} d_{m_1} &= \sqrt{(({}^{1}_{m_1}T[1,4])^2 + ({}^{1}_{m_1}T[2,4])^2)} = \sqrt{(0.25c_2^2l_3^2 + l_2^2)} \\ d_{m_2} &= \sqrt{(({}^{1}_{m_2}T[1,4])^2 + ({}^{1}_{m_2}T[2,4])^2)} = \sqrt{((0.5s_{23}l_4 - c_2l_3)^2 + l_2^2)} \\ d_{m_3} &= \sqrt{(({}^{1}_{m_3}T[1,4])^2 + ({}^{1}_{m_3}T[2,4])^2)} \\ &= \sqrt{((-((c_{23}c_4c_5 - s_{23}s_5)c_6 - c_{23}s_4s_6)l_5 + \\ &\quad s_{23}l_4 - c_2l_3)^2 + (-(s_4c_5c_6 + c_4s_6)l_5 + l_2)^2)} \end{aligned} \tag{4.36}$$

The corresponding mass of inertia for joint one is the sum of the individual ones and can be computed as:

$$I_1 = I_{m_1} + I_{m_2} + I_{m_3} = \frac{1}{2}(m_1d_{m_1}^2 + (m_2d_{m_2}^2 + (m_3d_{m_3}^2) \tag{4.37}$$

The torque acting on joint one, caused by the mass of inertia is:

$$\tau_{I1} = I_1\ \ddot{\Theta}_1 \tag{4.38}$$

4.4.2 Gravitational Effects

To compute the torque caused by mass gravity on each joint, the portion of each gravity vector which is in orthogonal direction to this joint has to be known. The difficulty is given by the serial kinematics, that means for example that the positions of all joint angles have to be regarded to compute the portion of the m_3 gravity vector acting orthogonal to joint five or six. One possibility to compute the joint torques, caused by static forces are with the help of Jacobian matrices.
The torques at each joint can be computed with the equation

$$\tau = J^T\ F \tag{4.39}$$

where τ is a 6 x 1 vector of torques at the joints, J is the 6 x 6 Jacobian with respect to the end-effector. F is a 6 x 1 cartesian force moment vector acting at the end-effector [10].

The advantage when dealing with gravity is that it works always in negative Z direction with respect to the base frame. This fact is used to write Equation 4.39 with respect to frame one as

$$\tau = {}^0 J^T \, {}^0F = \begin{bmatrix} {}^0J_{11} & {}^0J_{21} & \cdot\cdot & {}^0J_{61} \\ {}^0J_{12} & {}^0J_{22} & \cdot\cdot & {}^0J_{62} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ {}^0J_{16} & {}^0J_{26} & \cdot\cdot & {}^0J_{66} \end{bmatrix} = \begin{bmatrix} {}^0f_x \\ {}^0f_y \\ {}^0f_z \\ {}^0\tau_x \\ {}^0\tau_y \\ {}^0\tau_z \end{bmatrix} \tag{4.40}$$

The only component in the force moment vector is the gravity, so that Equation 4.40 changes to

$$\tau = {}^0 J^T \, {}^0F = \begin{bmatrix} {}^0J_{31} \\ {}^0J_{32} \\ {}^0J_{33} \\ {}^0J_{34} \\ {}^0J_{35} \\ {}^0J_{36} \end{bmatrix} [{}^0f_z] \tag{4.41}$$

This means that only the third row of the Jacobian (with respect to the base) is required. The Jacobians have to be computed for all three masses. Each of the Jacobian ${}^0_{m_1}J$, ${}^0_{m_2}J$ and ${}^0_{m_3}J$ transforms the gravity forces acting on the base frame into joint torques. To derive a Jacobian matrix, we consider the variables describing the position and rotation of a mass frame. All variables are functions of joint angles Θ_1 to Θ_6 and can be written as:

$$\begin{aligned} x_{m_i} &= f_1(\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6) \\ y_{m_i} &= f_2(\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6) \\ z_{m_i} &= f_3(\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6) \\ . & \quad . \\ . & \quad . \\ \gamma_{m_i} &= f_6(\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6) \end{aligned} \tag{4.42}$$

The Jacobian consists of the partial derivatives of these variables. Because we are only interested in the third row of the Jacobian, we compute

$$\delta z_{m_i} = \frac{\partial f_3}{\partial \Theta_1}\delta\Theta_1 + \frac{\partial f_3}{\partial \Theta_2}\delta\Theta_2 + ... + \frac{\partial f_3}{\partial \Theta_6}\delta\Theta_6 \tag{4.43}$$

or

$$\delta z_{m_i} =_{m_i} J_{31}\delta\Theta_1 +_{m_i} J_{32}\delta\Theta_2 + ... +_{m_i} J_{36}\delta\Theta_6 \tag{4.44}$$

z_{m_i} has to be derived from the transformation matrices, describing the transformation from base system to m_i:

$$\delta z_{m_i} =^0_{m_i} T[3,4] \tag{4.45}$$

Differentiating the z_{m_i}'s partially leads to three Jacobian matrices (only third row):

$$
{}^{0}_{m_1}J = \begin{bmatrix} 0 \\ 0.5c_2l_3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.46}
$$

$$
{}^{0}_{m_2}J = \begin{bmatrix} 0 \\ -0.5s_{23}l_4 + c_2l_3 \\ -0.5s_{23}l_4 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.47}
$$

$$
{}^{0}_{m_3}J = \begin{bmatrix} 0 \\ -((-c_{23}c_4c_5 + s_{23}s_5)c_6 + c_{23}s_4s_6)l_5 - s_{23}l_4 + c_2l_3 \\ -((-c_{23}c_4c_5 + s_{23}s_5)c_6 + c_{23}s_4s_6)l_5 - s_{23}l_4 \\ -(s_{23}s_4c_5c_6 + s_{23}c_4s_6)l_5 \\ -(s_{23}c_4s_5 - c_{23}c_5)c_6l_5 \\ -(-(-s_{23}c_4c_5 - c_{23}s_5)s_6 + s_{23}s_4c_6)l_5 \end{bmatrix} \tag{4.48}
$$

If each of the Jacobians and the corresponding gravity force are inserted in Equation 4.41, the torques can be computed as:

$$
\begin{aligned}
\tau_{m_1} &= {}^{0}_{m_1}J\ (-m_1g) \\
\tau_{m_2} &= {}^{0}_{m_2}J\ (-m_2g) \\
\tau_{m_3} &= {}^{0}_{m_3}J\ (-m_3g)
\end{aligned} \tag{4.49}
$$

Finally, the resulting torques for each joint are computed, after inserting Equations 4.46 to 4.49, as sum of the three individual ones:

$$
\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} = \tau_{m_1} + \tau_{m_2} + \tau_{m_3} \tag{4.50}
$$

Chapter 5

Controller Design

In this chapter, different control architectures are introduced, implemented on the testbed and experimental results are compared. Finally, the control stability is proved. Before designing a suitable controller, the demands have to be defined.

The controller will be designed for living up to the following conditions:

1. Fast and exact following of changing command values (input of a new reference position)
2. Robust behavior on parameter fluctuations (lifting of a load resp. movement of load)
3. Low over shooting ($< 10\%$)

To analyze if the controller fits the needs described above, the following experiments were attended:

a. Give a step input to the joint, see the response (Demand 1.)

b. Testing the ability to follow an oscillating reference value by giving a sinus curve to the controller input (Demand 1.)

c. Study the behavior on parameter fluctuations by giving a step input with different joint parameters, represented by attached load (Demand 2.)

Demand 3. is important for all types of experiments, but especially for experiment b.), because slow changing inputs will be the main case in the arm control.

Due to the high nonlinearity of the system combined with the serial kinematics, a fast and robust control is necessary to achieve the desired motion. According to the results in Section 4.2, a regular PID controller seems to be overstrained with handling these problems, thus special control techniques are required.

To reduce the error for the position control loop, caused by the nonlinearity and the hysteresis of the artificial muscles, the idea to implement a subsidiary torque control loop came up. This inner loop changes the system behavior that it appears to the outer control loop to be a linear element. Controlling the system on the level of torques has several advantages. Dynamic models of arm parts can be added if necessary for all joints or just to consider gravity and masses of inertia in the shoulder and upper arm joint. The torque control loop can be run separately, for a passive arm-movement (e.g. shaking hands). Testing of the inner control loop is possible independently from the rest of the control and allows easier parameter tuning.

For this reason, an approach to improve the conventional PID control by implementing a cascaded controller with a inner torque control loop is followed. To judge the performances of both control types, experiments have been run on a testbed and the results are discussed.

As a first architecture, a standard PID controller was implemented on the testbed as described in Section 5.1.

The second architecture is a cascaded PI-PI controller, with separate control loops for position and torque. Section 5.2 explains a possible implementation with a inner PI torque controller and a outer PI position controller. Torque or force sensors are required to realize the feedback control.

To realize the previous architecture without using torque or force sensors in the joint, an actuator model (as developed in Section 3) is used as feedback to predict the actual joint torque. Accurate actuator modeling is necessary to receive similar control results as above. This architecture will be introduced in Section 5.3.

In the last architecture, described in Section 5.4, the cascaded model-based control was extended with an nonlinear part to compensate the most disturbing factors like gravity and masses of inertia.

5.1 PID controller (PID)

5.1.1 PID controller implementation

Figure 5.1 shows the implementation of a regular PID controller. The system consists of pressure control unit, actuator and arm. The controller output sets directly Δp in the pressure control unit. The only sensory information are encoder steps, from whose the joint velocity is derived.

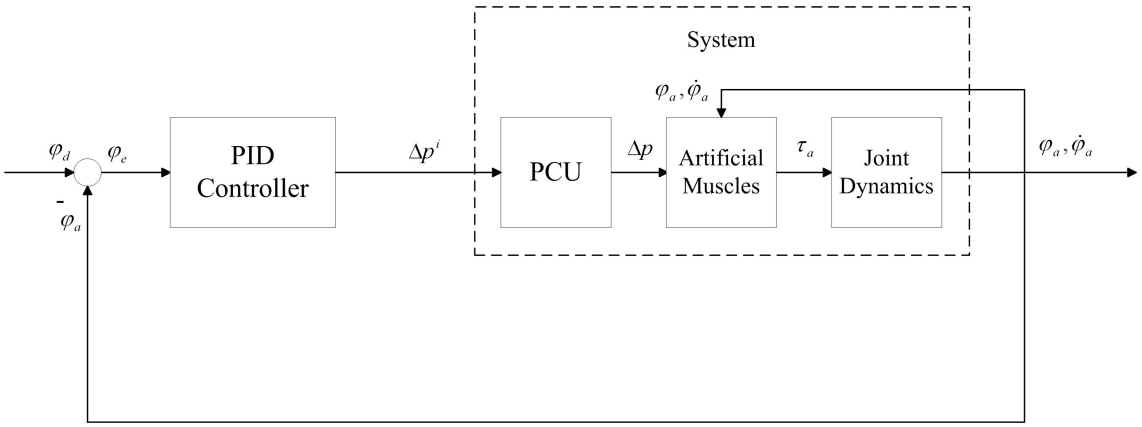

Figure 5.1: Control architecture using PID controller

The transfer function for PID controller is given by:

$$F_{PID}(s) = K_P + \frac{K_I}{s} + K_D s \tag{5.1}$$

The experiments were run under QNX with a sample rate of 2 kHz. To tune the PID parameters, two different methods, Ziegler-Nichols method (ZN) and Chien, Hrones and Reswick method (CHR) were applied.

The Ziegler-Nichols is a common method to tune PID parameters in process engineering where usually much higher time constants exist, but gave also satisfying results for this

system. After increasing the proportional gain until a undamped oscillation occurred, the parameters were determined as $K_{P,crit}$=1.25 s and T_{crit}=0.175 s. According to the Ziegler-Nichols approach, the control parameters have to be set as

$$K_P = 0.6 \cdot K_{p,crit} = 0,75,\ K_I = \frac{K_p}{0.5 T_{crit}} = 14.3 \text{ and } K_D = 0.12 \cdot K_P \cdot T_{crit} = 0.016.$$

Due to instable behavior of the joint, the gain had to be reduced to $K_P = 0.4$ to guarantee stable movement. Therefore, other control parameters changed to $K_I = 7$ and $K_D = 0.08$.

The Chien, Rhones and Reswick method is based on Ziegler-Nichols, but improved on basis of simulation results. Control parameter tuning can be designed according to certain criteria, depending if the control is to be optimized for fast following of reference values, or for fast control of occurring disturbances. Experiments with both criteria showed that designing the controller for fast following of reference values gives better results to satisfy the overall controller demands.

The tuning of this method is based on system parameters latency T_L and adjustment time T_A. These parameters are determined as described in Section 4.2. The control parameters can be computed as

$$K_P = \frac{0.6 T_A}{K_J T_L} = 6,\ K_I = \frac{K_P}{T_A} = 120 \text{ and } K_D = 0.5 \cdot K_P \cdot T_L = 0.04.$$

The same instability problem like above required tuning down the proportional gain to $K_P = 0.2$, which decreased the other parameters to $K_I = 4.0$ and $K_D = 0.0014$.

5.1.2 Experimental results of PID controller

In the first experiment (Figure 5.2), three step inputs were given to the controller. The Chien, Hrones and Reswick method showed a smoother, but slower reaching of the reference value. Fine-tuning of the Ziegler-Nichols parameters did not allow to get rid of the small spike.

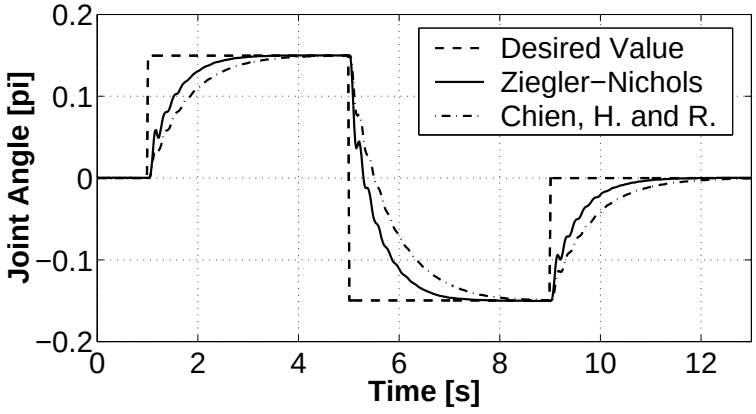

Figure 5.2: PID controlled step response

The next experiment shows the ability to follow a slow changing reference value, represented by a sin curve $f(t) = 0.15\ sin(2t)$. Both parameter sets could not follow this trajectory satisfying, the difference was up to 18° (CHR tuning) between reference position and joint position.

Figure 5.4 shows the step response of the joint with different parameters. In this case, a mass of 0.8 kg was added to the arm to change the mass of inertia from J=0.0085 kgm^2 to

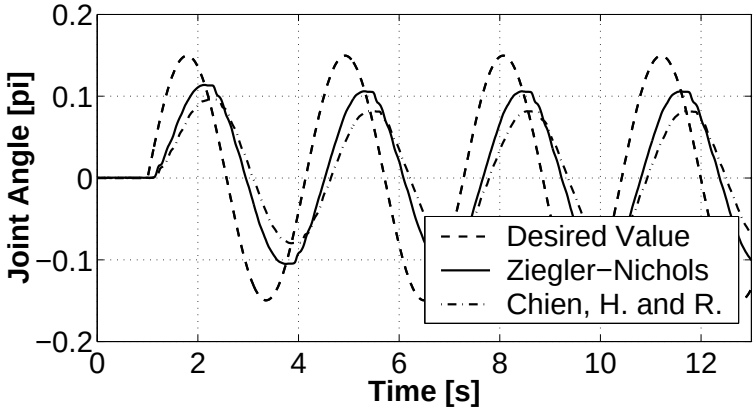

Figure 5.3: PID controlled joint response on oscillating controller input

approximately J=0.036 kgm^2. The joint response was an oscillating movement after giving a step input as new reference value. These oscillations could be minimized by decreasing the proportional gain, but the adjustment time was increasing and stood in no relation to the smoother curve.

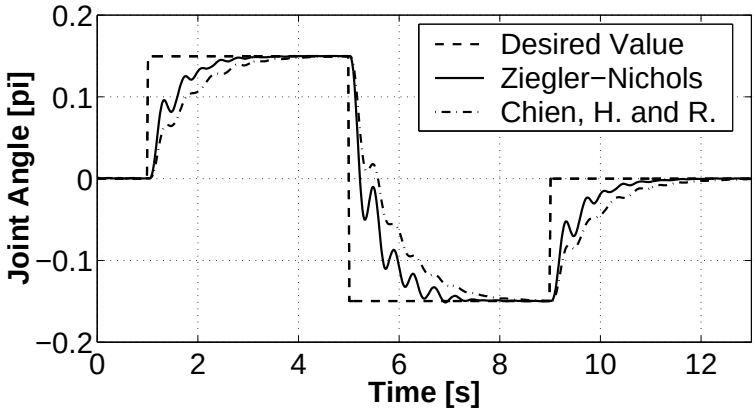

Figure 5.4: PID controlled step response after changing mass of inertia

The response on a occurring disturbance (deflection and fast release by hand) shows Figure 5.5. The arm reaches the starting position after several oscillations (after 0.8-1.0 s). The overshoot was with 27° very high, compared to a deflection of only 18°.

5.1.3 Stability of PID controller

To prove stability of the control, the characteristic formula is derived:

$$1 + F_{PID} F_C F_J = 0 \tag{5.2}$$

Where F_{PID} is the controller transfer function, F_C and F_J are transfer functions of pressure control unit and joint. Inserting the transfer function leads to

$$0.017s^4 + 0.4s^3 + (30K_D + 18.5)s^2 + (30K_P + 100)s + 30K_I = 0 \tag{5.3}$$

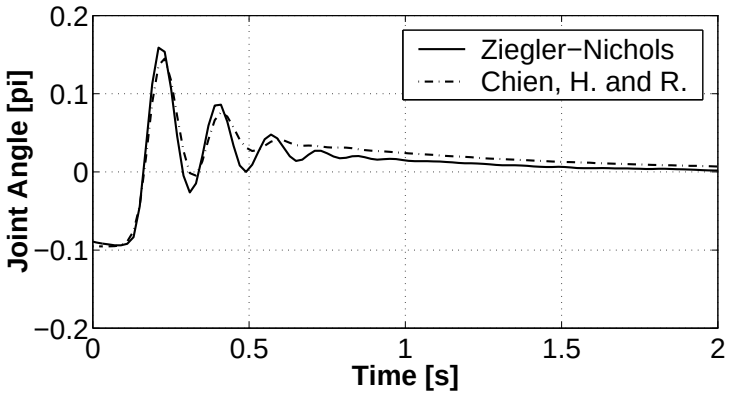

Figure 5.5: PID controlled joint response on occurring disturbances

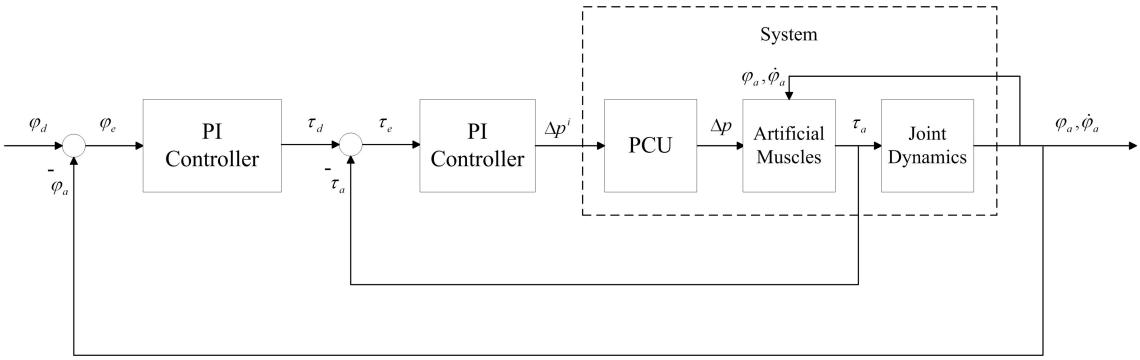

Figure 5.6: Cascaded controller architecture

Because all coefficients are positive, it depends on the last demand of the Hurwitz-Criterion to prove absolute stability, which is

$$a_1 a_2 a_3 - {a_1}^2 a_4 - a_0 {a_3}^2 > 0 \tag{5.4}$$

The result after inserting the coefficients is $6.8 > 0$ for the Ziegler-Nichols parameter set and $5.7 > 0$ for the parameters derived from the Chien, H. and R. method, which proves stability for both approaches.

5.2 Cascaded controller (PI-PI)

A possible implementation of a cascaded controller with different control loops for torque and position is shown in Figure 5.6. The idea is to give the inner control loop a fast time response, so that the behavior is similar to a linear transfer element for the outer control loop. The sample times of inner loop to outer loop are chosen as 1:10 to provide the required time for inner loop to adjust the reference value.

The torque controller is implemented as a PI controller. The I-part is needed so that disturbances of the inner loop are controlled in order that the loop can be run in stand-alone torque control mode. A D-part is not necessary because of the high sample rate of the inner loop and would complicate the parameter tuning. Experiments showed that tuning of more than four parameters is very difficult and the results were rather stability problems than

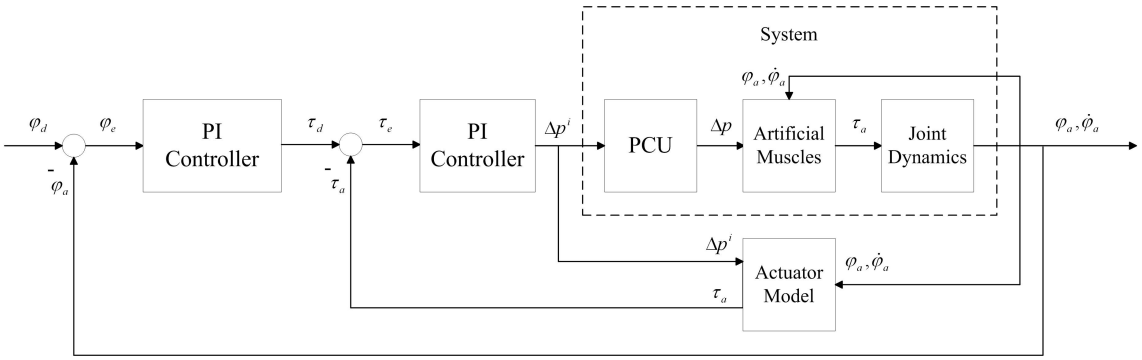

Figure 5.7: Model-based cascaded controller architecture

improved control speed. Because of this fact, the outer controller was also realized as PI controller without D-part. The dynamic of the control is a outcome of the controller architecture itself, not a separate differentiating part.

The transfer functions of torque and position controllers are

$$F_{C\tau} = \frac{1}{K_{P\tau} + \frac{K_{I\tau}}{s}} \tag{5.5}$$

$$F_{Cp} = \frac{1}{K_{Pp} + \frac{K_{Ip}}{s}} \tag{5.6}$$

To minimize cost and hardware effort, no force or torque sensors are used in the ISAC arm. To realize a closed-loop control, torque feedback is required. This means the architecture in Figure 5.6 cannot be implemented in this way. The following section introduces another possibility to implement a cascaded control with torque feedback.

5.3 Model-based cascaded controller (PI-MBPI)

5.3.1 Model-based cascaded controller implementation

One approach to realize the cascaded control architecture without measuring torque is shown in Figure 5.7. The torque feedback is provided by our actuator model, described in Chapter 3. The fast torque control loop runs at a sample rate of 2 kHz, the slower position control loop with a rate of 0.2 kHz.

First step was the implementation of the torque control, and tuning the inner PI parameters. For this purpose, the joint was fixed in its position so that no movement was allowed. The parameters were adjusted as $K_{P,T} = 0.1$ and $K_{I,T} = 10$ by Ziegler-Nichols, which led immediately to good results. After tuning the inner loop parameters, the torque controller was tested separately. Figure 5.8 shows a response of the torque controller for different step inputs.

The inner control loop has now a behavior like a first order lag time element with a time constant $T_T = 0.025$. An important fact is that the adjustment time hardly depends on the amount of the reference torque. The torque was reached after 0.018 s to 0.02 s, little differences are caused by different filling times but do not need to be considered.

None of the common tuning methods seemed to work for the position control loop, this might have to do with the cascaded control structure and the limitations of torque and

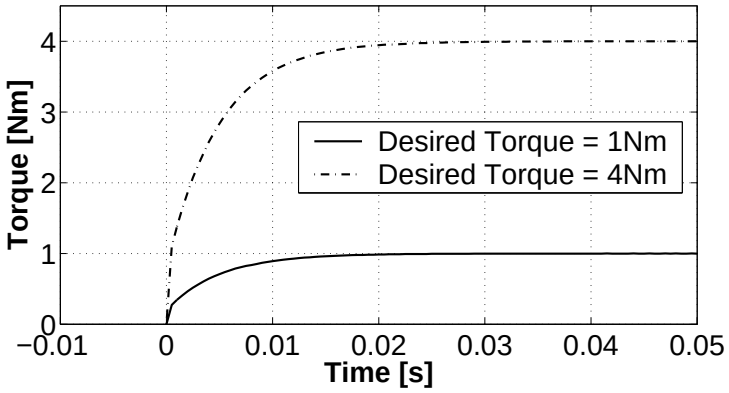

Figure 5.8: Step response of inner torque control loop

pressure in the controller outputs. Therefore, outer loop parameters were adjusted by trial and error, starting with low $K_{P,P}$, increasing the proportional gain and adding the integral gain later. The outer loop control parameters were identified as $K_{P,P} = 4.0$ and $K_{I,P} = 8.0$.

5.3.2 Experimental results of model-based cascaded controller

The joint response on a step input (Experiment a.), Figure 5.9) was now faster and smoother than with a regular PID controller. Much more important is the response on the sinus curve (Experiment b.)), given in Figure 5.10. The joint position follows the reference value nearly without lag. Due to the fact that the joint has to follow a trajectory in future use on ISAC's arm, this is the biggest improvement compared to the PID controller.

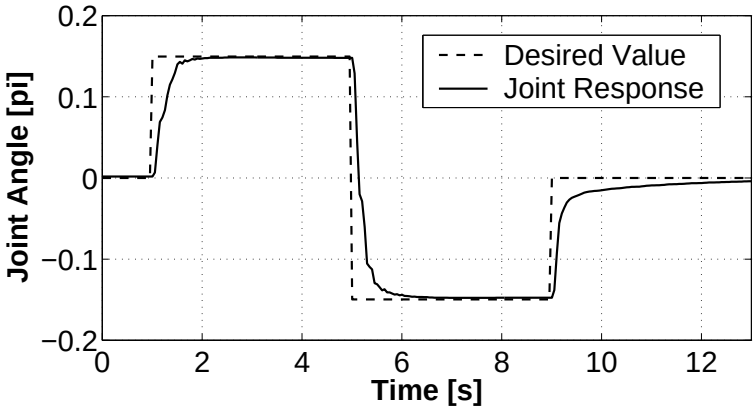

Figure 5.9: PI-MBPI controlled step response

Change of parameters (Experiment c.), Figure 5.11) produces an overshoot of 10-15 %. This overshoot can be accepted because the controller input will never be a step of this height. Experiments with higher mass of inertia and the sinus trajectory in Experiment c.) showed that the overshot does not play any role when the controller input is a sequence of very small steps.

The control of disturbances is also improved compared to the regular PID controller. The PI-MBPI controller has an overshoot when the arm of the joint is pushed and released, but the overshoot is not as high (only 50 %) and the adjustment time to reach the initial position

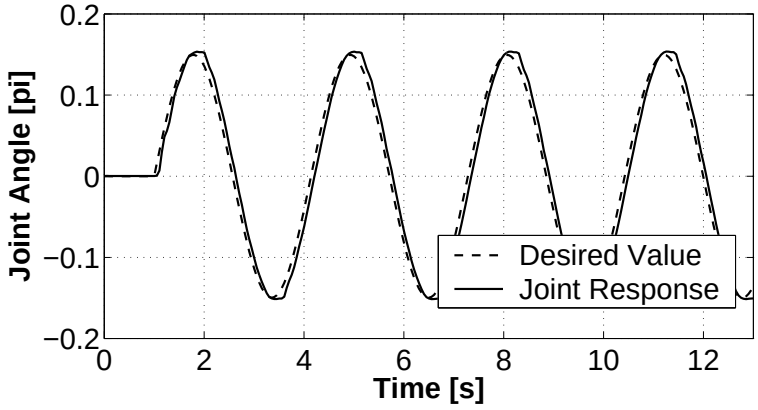

Figure 5.10: PI-MBPI controlled joint response on oscillating controller input

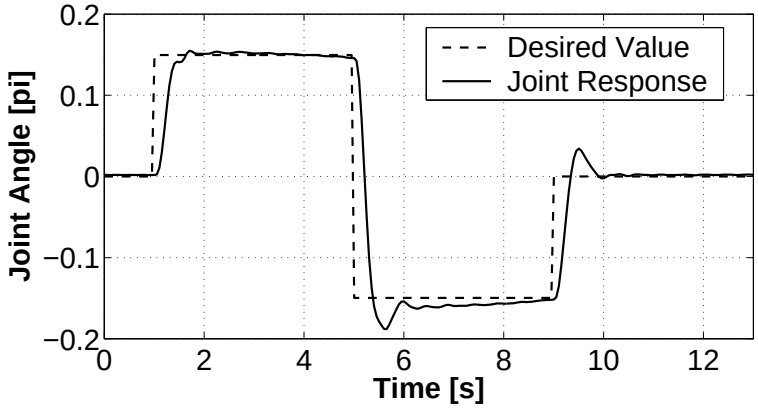

Figure 5.11: PI-MBPI controlled joint response after changing mass of inertia

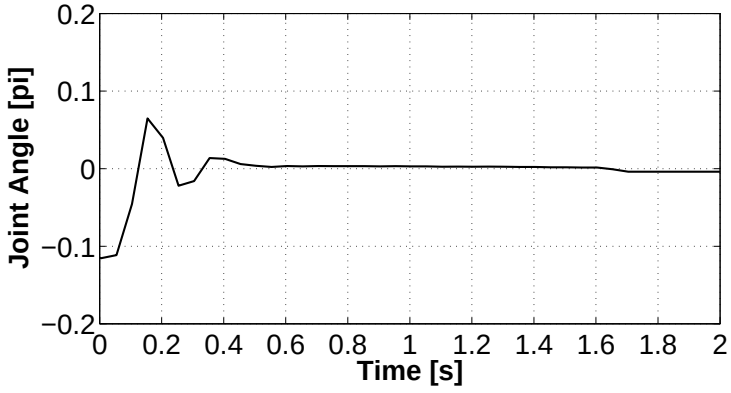

Figure 5.12: PI-MBPI controlled joint response on occuring disturbances

is with 0.5 s less.

5.3.3 Stability of model-based cascaded controller

To prove control stability, the closed-loop transfer function of the system has to be derived. Instead of using the transfer function of the model-based controller, the easier architecture in Figure 5.6 is used. Provided that the actuator model is close to the real behavior, this assumption is allowed. The damping parts of the system have been ignored, that means a simplified version of the joint transfer function is used for the proof of stability. This is possible because the damping part would take energy out of the system and so increase stability. Transfer functions of the different components are listed in Table 5.3.3.

Pressure Control Unit	$F_C(s) = \frac{1}{1+T_C s}$
Torque Controller	$F_{C\tau}(s) = \frac{1}{K_{P\tau}+\frac{K_{I\tau}}{s}}$
Position Controller	$F_{Cp}(s) = \frac{1}{K_{Pp}+\frac{K_{Ip}}{s}}$
Joint (Actuator and Dynamics)	$F_J(s) = \frac{Br}{Js^2-Ar^2}$
Torque Feedback	$\tau_a = Ar^2\varphi_a + Br\Delta p^i$

Table 5.1: System transfer functions

Because all transfer functions are linear, which means that our overall transfer function will be a linear equation, the proof of stability can be done with the Hurwitz criterion [16]. The idea is to locate the poles of the linear closed-loop function in the s-plane. All poles have to lie in the left half of the $s - j\omega$ diagram to prove stability. Therefore, coefficients of the characteristic equation have to be analyzed.
The closed loop transfer function is:

$$\frac{\varphi_a}{\varphi_d} = \frac{F_0(s)}{1 + F_0(s)} \tag{5.7}$$

To determine poles of the system, the denominator has to be zero:

$$1 + F_0(s) = 0 \tag{5.8}$$

The open loop transfer function $F_0(s)$ is derived by summarizing the individual transfer functions and solving the links caused by the actuator model with three independent variables as

$$F_0(s) = \frac{Br(K_{P\tau} + \frac{K_{I\tau}}{s})(K_{Pp} + \frac{K_{I\tau}}{s})}{((Js^2 - Ar^2)(1 + T_C s)) + Js^2 Br(K_{P\tau} + \frac{K_{I\tau}}{s})} \tag{5.9}$$

Insertion in Equation 5.8 leads to the characteristic formula

$$\begin{aligned}(JT_C)s^5 + (J + JK_{P\tau}Br)s^4 + (JBrK_{I\tau} - Ar^2T_C)s^3 + \\ (BrK_{Pp}K_{P\tau} - Ar^2)s^2 + (BrK_{Pp}K_{I\tau} + K_{Ip}K_{Pp})s + \\ (BrK_{Ip}K_{I\tau}) = 0\end{aligned} \tag{5.10}$$

Comparison with the regular equation

$$a_0 s^5 + a_1 s^4 + a_2 s^3 + a_3 s^4 + a_4 s^5 + a_5 = 0 \tag{5.11}$$

leads to the coefficients $a_0 - a_5$. The first condition of the Hurwitz criterion says that all coefficients have to be positive. Considering the fact that the parameter A is always negative, and all others are positive, this condition is fulfilled. The second condition demands that all determinants $H_2..H_5$ are positive. H_i are $[i, i]$ sub-determinants derived from the Hurwitz determinant starting at the upper left corner.
The Hurwitz determinant for our system is

$$H = \begin{vmatrix} a_1 & a_3 & a_5 & 0 & 0 \\ a_0 & a_2 & a_4 & 0 & 0 \\ 0 & a_1 & a_3 & a_5 & 0 \\ 0 & a_0 & a_2 & a_4 & 0 \\ 0 & 0 & a_1 & a_3 & a_5 \end{vmatrix} = \begin{vmatrix} 0.011 & 9.5 & 200 & 0 & 0 \\ 0.0015 & 1.7 & 100 & 0 & 0 \\ 0 & 0.011 & 9.5 & 200 & 0 \\ 0 & 0.015 & 1.66 & 100 & 0 \\ 0 & 0 & 0.011 & 9.5 & 200 \end{vmatrix} \tag{5.12}$$

Computing the determinants H_2-H_5 gives as result:

$H_2 = 0.0040$, $H_3 = 0.030$, $H_4 = 5,4$ and $H_5 = 1100$

That means that all poles of the system are lying on the left side of the imaginary axis. The system is hereby always stable. Stability is also given for higher values of J. Changing the mass of inertia to J=0.036 kgm^2 to simulate a possible load even stabilizes the system more by increasing the determinants.

On the basis of the received results, the cascaded model-based controller (PI-MBPI) is preferred instead of using a regular PID control. In particular the fast following of reference values and the handling of parameter changes are the deciding features. Practical reasons, like easy extension upon dynamic models and the ability to run the arm in torque-control-mode, support this choice.

It was shown that the realization of a torque control loop is possible without using any torque or force sensors. The feedback can be given by an actuator model, which reduces the required hardware and with it costs.

5.4 Extending the model-based cascaded controller (PI-CMBPI)

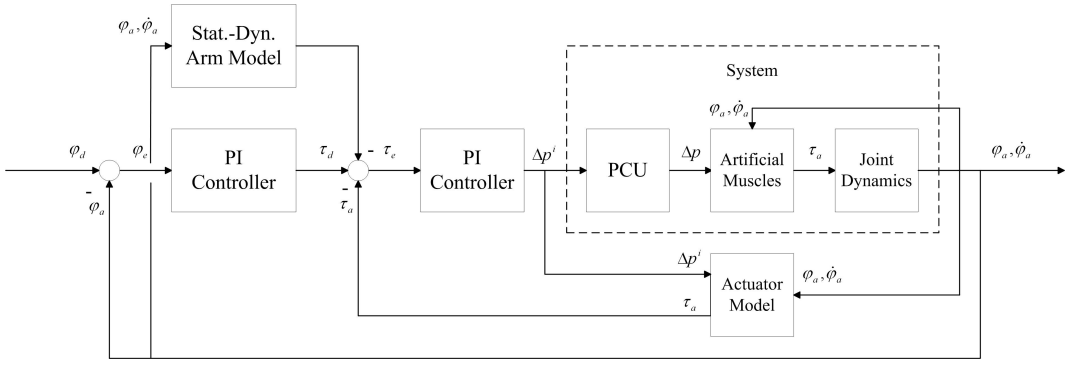

Figure 5.13: Extended model-based controller architecture

To be able to compensate torques caused by gravity and masses of inertia, another part was added to the torque control loop according to Figure 5.12.

This part is able to compensate specific torques for each joint. Because a complete development of a dynamic arm model is a big effort, only gravitational and inertia effects have been considered for some joints to improve the control with low effort as described in Section 4.4. For now, compensation of torques for joints which are exposed to serious effects is implemented. These are gravity effects for shoulder and elbow joint and inertia effects for the shoulder joint turning the entire arm. Experimental results are presented in Chapter 7, no experiments were performed on the testbed due to the missing complexity of the kinematics.

Chapter 6

Control Implementation on ISAC

This chapter describes the identification of actuator parameters, gives an overview over the control realization in software, tuning of control parameters and identification of compensation masses m_1, m_2 and m_3.

6.1 Identification of actuator parameters

To be able to run the control on ISAC, the different model parameters have to be set first. ISAC uses four different types of muscles, what means that we have four different actuator models and 16 variables to identify. For this purpose, the different types were mounted in the testbed. Quasi-static measurements according to experiments illustrated in Figure 3.7, and dynamic experiments according to Figures 3.8 and 3.9 were run to identify the parameters A, B, C, and D. Because the diameter of the chain wheel is different for each joint, it was considered separately in the equation and has also to be measured.

The actuator parameters were identified as following:

Joint No.	A $[\frac{N}{m}]$	B $[N\frac{cm^2}{kg}]$	C $[Nms^3]$	D $[Nms]$	Diameter $[mm]$
1	-143000.0	3200.0	0.0351	2.20	0.030
2	-108000.0	970.0	0.0184	1.14	0.025
3	-80600.0	334.0	0.0046	0.286	0.020
4	-80600.0	329.0	0.0046	0.286	0.020
5	-37600.0	138.0	0.0023	0.143	0.015
6	-37600.0	134.0	0.0023	0.143	0.015

Table 6.1: Actuator model parameters

A critical point is the muscle transfer from the testbed to ISAC. A change of the muscles pre-stressing would result in different parameters, and would cause an error which cannot be corrected later. Therefore, it is very important to mount the Shadow muscles, which have a maximum contraction ratio of 30 %, carefully with a pre-stressing of 15 %. The Bridgestone, which have a maximum contraction ratio of about 20 % should only be mounted with 10% pre-stressing.

6.2 Control software overview

The control for ISAC is written in C++ and runs on a QNX platform. This real time operating system allows a sample rate of maximal 2 kHz. Not much other and no time-critical applications are running on this machine beside the control. Therefore, the complete inner torque control, together with input/output routines, can be run with a high sample rate of 2 kHz. The slower position control loop runs with a sample rate of 0.2 kHz. A higher sample rate would not lead to better results because the inner torque control loop needs a couple of cycles to be able to adjust the desired torque before it gets a new input.

The control architecture developed in Chapter 5 is realized separate for each joint. This means that the functions containing the control algorithms have to be run for all six joint instances. Figure 6.1 gives an overview over the most important modules and the C++ functions representing them.

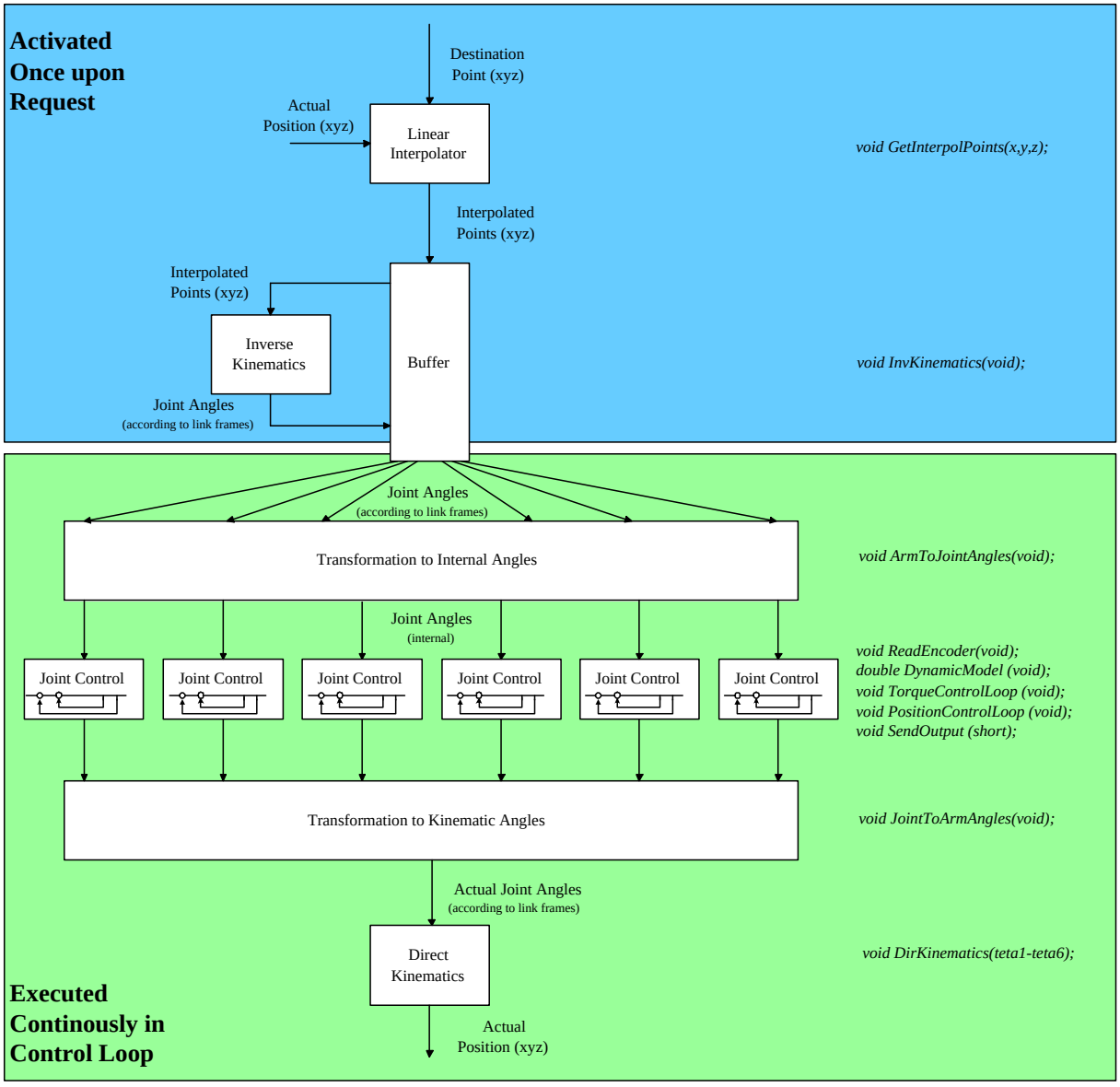

Figure 6.1: Software overview

The position controllers for each joint have to be fed with desired joint positions. Because it makes no sense to give the joint positions for each joint directly, the inverse kinematics are used to compute joint angles, corresponding to a XYZ position. To guarantee a smooth movement, the trajectory consists of lots of cartesian points which have a small distance. This leads to joint angles with small distance so that no step input is given to the controller and over shooting can be kept low.

At this time, no complex trajectory generator was implemented yet, so a simple generator computing a linear trajectory to a destination point was programmed. Once a new destination point is set, interpolated points in XYZ coordinates, starting at the actual position, are computed and stored in a buffer. The Euclidean distance between these points is the interpolating distance, set to 2 mm for this generator. The inverse kinematics compute, if possible, a set of joint angles corresponding to the XYZ positions and store them again in the buffer. The check whether the destination point is reachable or not is also performed here, this is not possible at an earlier state because joints limitations can only be considered after the inverse kinematics.

Moving the arm directly to a destination point is not a very smart and smooth movement. The XYZ input to the buffer can be more complex trajectories, what will be the case in future. The control does not care what the input is, as long as the trajectory points have a distance from each other of 2 mm's or lower.

Once the joint angles are set in the buffer, the control picks periodically a new set of desired joint angles, transforms them into internal joint angles and gives them as input to the control. The kinematic angles θ_1 to θ_6 are the joint positions according to the link frames attached to the arm in Figure 4.3. Internal joint angles φ_1 to φ_6 are in the range of $\pm\pi$, and refer to the middle position of the actuator, where the muscles have both the same length. This position corresponds to $\Delta p = 0$ in unloaded state.

The task of the control loop (run with sample rate of 2kHz) can approximately be summarized as:

- Reading encoder angles
- Computing actual torque with actuator model
- Computing static and dynamic parts to compensate gravitational and inertia effects
- Run torque control loop
- Run position control loop (only every 10 times, 0.2 kHz)
- Send output to the servo valves

The control picks variables out of the buffer as long as there are desired joint angles available. The sampling rate defines the velocity of the arm. To move the arm with a velocity of v $[\frac{m}{s}]$, the sample rate for giving new inputs has to be $f_v = \frac{v}{s_i}$ in $[\frac{1}{s}]$, where s_i $[m]$ is the Euclidean distance between two interpolation points, 2 mm for this trajectory generator.

After transforming the internal angles back into kinematic angels, the XYZ position of the tool center point can be computed by using the direct kinematics.

A detailed description of all C++ functions can be found in Appendix D.2.

6.3 Tuning control parameters

Tuning control parameters in the system is much more difficult than in the testbed. The higher masses make the mechanics much more vulnerable, so that parameter tuning has to be carried out very carefully to avoid breaking parts of the arm.

To reduce the complexity and to get a better feeling for the influence of control parameters, the tuning was done for each joint after another, starting with the inner torque control loop first. Four parameters have to be tuned for each joint, two proportional gains and two integral gains.

The inner torque control loop is supposed to be as fast as possible, to guarantee a fast reaching of the desired value, before the position controller sends a new one. Therefore, only the P-part was increased from zero as high as possible to guarantee fast control but not too high to remain the system stable. The I-part was tuned afterwards until the torque controller had a overshoot of approximately 10 %. The result of the torque controller is limited per software to $\pm$ 2 $\frac{kg}{cm^2}$. This is not necessary to protect the muscles because the output pressure of the valves is limited anyway, but to keep the controller value down and prevent computing a wrong error and to increasing the integral part too much.

For the same reason, the limits of the position control output have been set according to the maximum values the torque controller can adjust. Tuning the position controller was nearly a similar procedure. The torque compensation part was disabled for this tuning to guarantee the functionality of the control even without this feature, and also to keep the architecture as simple as possible for the tuning process. Table 6.3 shows the resulting controller gains for each joint:

Joint No.	Proportional Gain Outer Loop	Integral Gain Outer Loop	Proportional Gain Inner Loop	Integral Gain Inner Loop
1	200.0	80.0	0.0065	10.0
2	100.0	25.0	0.03	10.0
3	25.0	10.0	0.12	5.0
4	25.0	10.0	0.12	5.0
5	8.0	2.0	0.4	10.0
6	8.0	2.0	0.4	10.0

Table 6.2: Control parameters

Usually, the control is run as a position controller by giving desired joint angles as input. This architecture allows also to run only the torque controller which can be an interesting feature. By setting the torque control mode active, the result of the position controller is cut off and a desired torque of 0 Nm is chosen as input for the torque controller. This allows moving the arm by hand, e.g. to teach movements or to shake hands with a human. It is imaginable to run the torque control mode only for certain joints, like for example the elbow and wrist part and to give a defined torque, which would make more sense for a hand shaking action.

6.4 Identification of compensation masses m_1, m_2, m_3

To compute torques caused by static and dynamic effects, the values of the three masses have to be identified. In the unloaded state, all masses represent the own weight of the arm. If a load is attached, mass m_3 can be increased. Since it takes too much effort to take the arm apart and to measure the masses, or the weight of the arm parts directly, they have to be identified by measuring the torques acting on the joints. It is obvious to use the actuator model for this task. The easiest way would be to move the arm in straight position and to let the actuator model predict the torque for each joint to compute the masses.

Because the shoulder joint cannot be extend to this position, and using the actuator model near the joint limitations should be avoided (compare actuator model restrictions in

Section 3.5), 45° in relation to the horizontal axis have been chosen for this joint, so that the state of the arm was: $\theta_1 = 0°$, $\theta_2 = -45°$, $\theta_3 = -45°$, $\theta_4 = 0°$, $\theta_5 = 90°$ and $\theta_6 = 0°$ (compare kinematic frames in Figure 4.3).

The masses are now causing torques on joints #2, #3 and #5, and have been, with the corresponding lever arms, identified as $m_1 = 1.750$ kg, $m_2 = 0.450$ kg and $m_3 = 0.140$ kg.

To verify the obtained values, the arm was now run in the torque control mode, and has been moved by hand to see if the gravity caused by masses is compensated. In opposite to not considering the gravity or to setting the masses zero, the arm could now stand in the air and did not move any more under the influence of gravity. Extension of the arm to joint limits caused the joints to push back a little bit. This is due to the lack of actuator modeling near the limits.

Chapter 7

Experimental Results

This chapter presents the results of final experiments performed on ISAC. The influence of the torque compensation part was tested on the PI-CMBPI controller and compared with results of a simple PI-MBPI controller. The response on parameter changes was analyzed by moving the arm with and without load, and the reaction of the arm on occurring disturbances was researched.

7.1 Influence of torque compensation part on the control

To see whether the extended architecture of the model-based cascaded controller leads to improvement of the control, it was compared in the following experiment with a regular model-based cascaded controller without compensation part. As described in Section 5.4, the torque caused by gravity as well as the torque caused by inertia effects is compensated in the architecture shown in Figure 5.13.

Two experiments offer themselves to verify the improvement of this compensation:

- Movement of the arm in vertical position (high influence of gravity) with and without compensation of torques
- Fast movement of axis number one with extended arm (high influence of inertia effects) with and without compensation of torques

7.1.1 Influence of gravity effects

To see if the gravity compensation part leads to higher control quality, the arm was moved along the Z axis a couple of times up and down. The speed of the tool center point was adjusted to 0.2 $\frac{m}{s}$. In Figure 7.1, the Z coordinate is plotted over time to see the deviation of movement with and without gravitational compensation. This experiment was performed without load.
Both curves contain time delays and do not differ much from each other for the uprising movement. The PI-MBPI controller without compensation seems to start a little bit slower than the PI-CMBPI controller. Considering the fact that the PI-CMBPI applies more torque to the joint for uprising movements, this is surprising, but checking the pressure values computed by the torque controller shows that the maximum limit is reached anyway, so it makes no more difference if the desired value is increased or not.

Looking at the down rising part shows, in contrast, a oscillation for the PI-MBPI controller and a smoother movement with PI-CMBPI controller which reduces the torque when moving

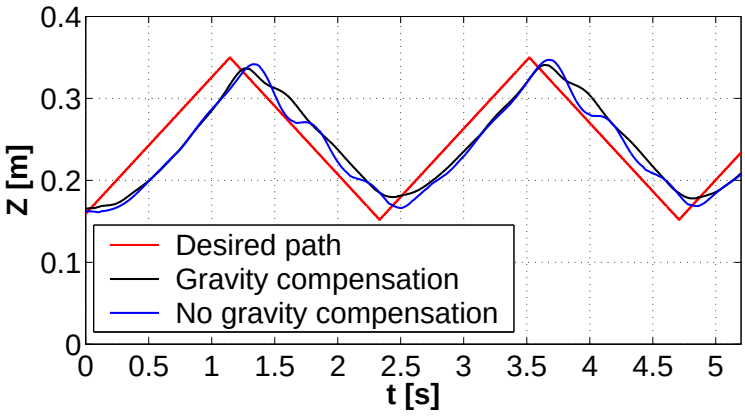

Figure 7.1: Gravity effects when moving arm along Z axis

downwards. The compensation of gravity seems to be more important for the movement in gravity direction. This can be explained with the fact that the regular controller gives an output which is too high, and the immediate following correction causes the oscillation. The PI-CMBPI controller reduces the input for the torque controller and prevents so oscillation because of a lower manipulated variable.

7.1.2 Influence of inertia effects

The inertia effects have only been considered for the first joint. The effect is largest if all masses have the maximum distance from joint one axis, so the arm was stretched as far as possible for this experiment. No additional load was attached. Joint number one was moved to a desired position and back a couple of times. The speed was 0.9 $\frac{rad}{s}$ for this joint, all other joints remained fixed. Figure 7.2 shows the position of the first joint over time.

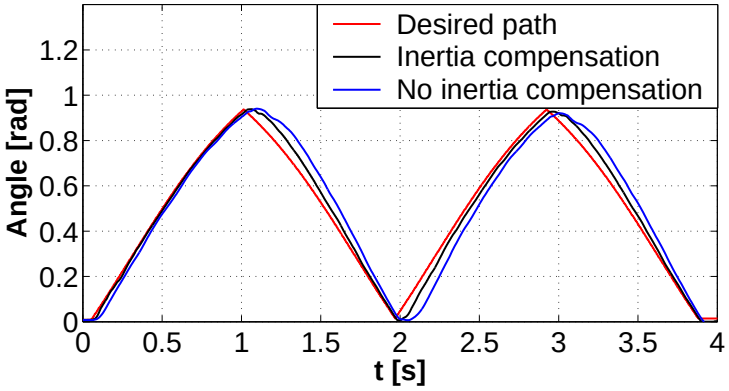

Figure 7.2: Inertia effects when moving arm around first joint

As can be seen in the figure, the error after a change of direction is lower if compensating the inertia effects and higher if using the regular PI-MBPI controller. The overall advantage of the PI-CMBPI controller is not so large, compared with the previous experiment. This might have to do with the fact that the acceleration (which causes the inertia effect) for this joint is compared to other joints relatively low due to the larger muscles and the longer filling time when applying pressure.

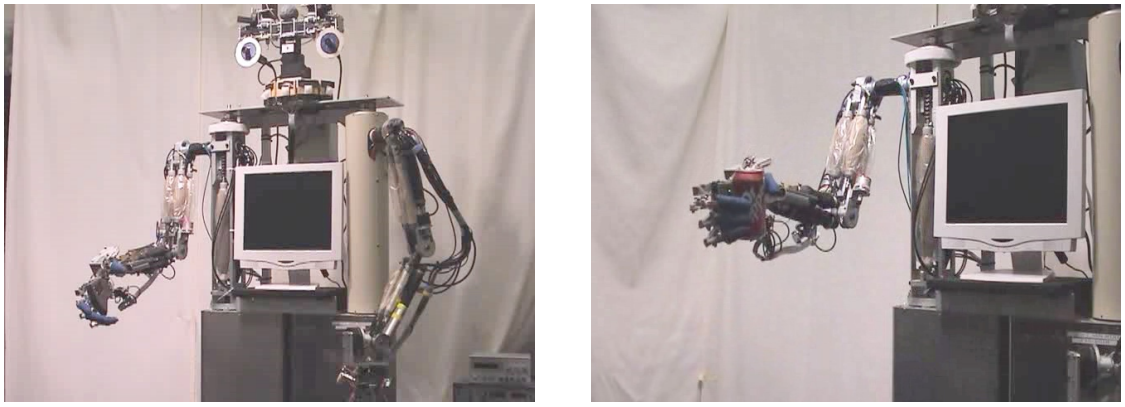

Figure 7.3: Experiments with plain hand and attached load

7.2 Simulating parameter changes when following a trajectory

After testing some joints and components intensively, this experiment shows the result of a regular movement and the influence of parameter changes. For this purpose, a triangle trajectory was chosen as controller input by giving three destination points. This is a very simple geometric form and makes it easy to judge the arms behavior by vision and also in the diagram. The parameter changes are realized by carrying a Dr. Pepper can (approximately 340 g weight), this is a situation that occurs often in real life (Figure 7.3).

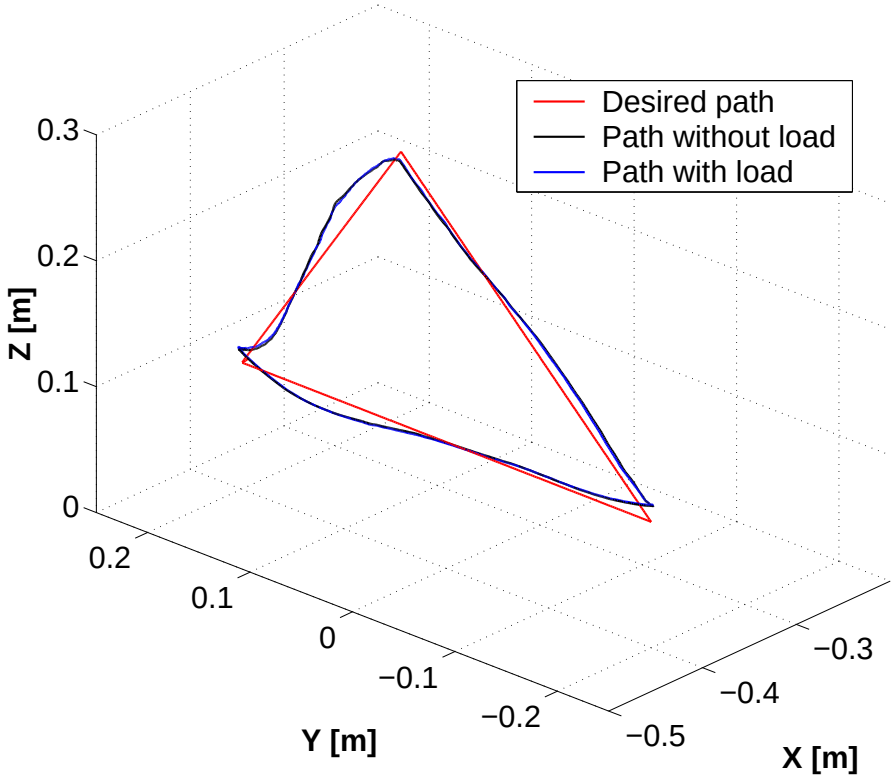

Figure 7.4: Arm following triangle trajectory

The triangle was tried to follow with a speed of 0.2 $\frac{m}{s}$ of the tool center point. Figure 7.4

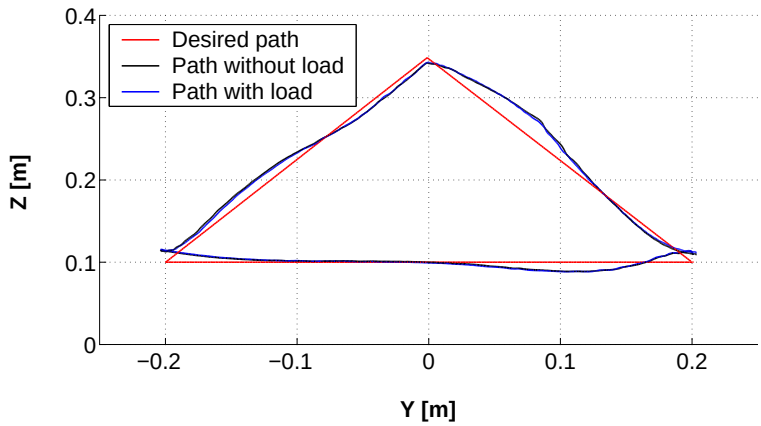

Figure 7.5: Arm following a triangle trajectory - front view

shows a 3D view of the experiment, a front view in Y-Z plane can be seen in Figure 7.5. The second Figure shows that there is no significant difference between the movement with load and the movement without load. These results will definitely change for higher loads, but proofs that the control can handle loads of about 350 g without recognizing any difference. Over shoots appear after a change of direction for both movements with and without load in the same way.

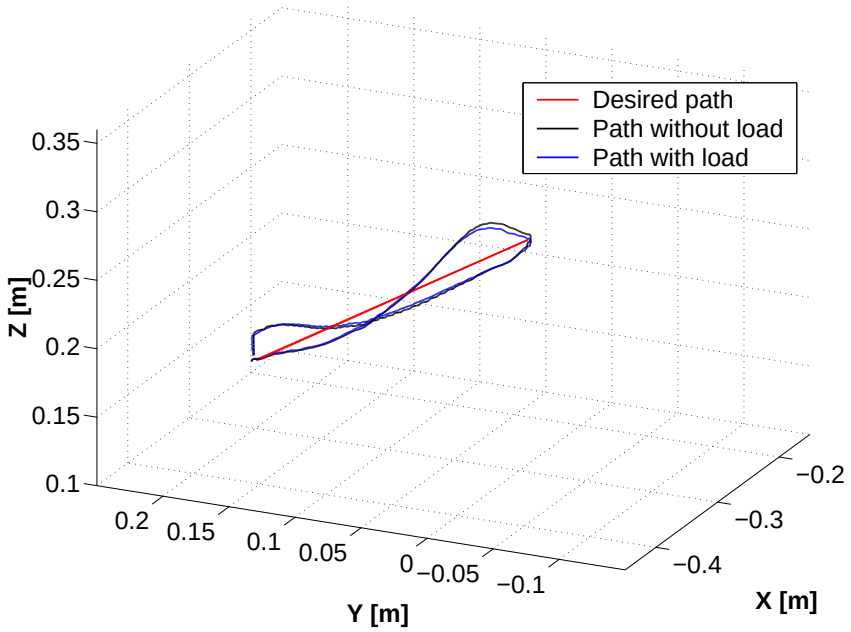

Figure 7.6: Reaching to point

The following Figures 7.6 and 7.7 show a fast movement to a destination point and back. The velocity of the tool center point was here 0.4 $\frac{m}{s}$, what lead to a heavy overshoot at the ends. The reason for this large error is that the servo vales are too slow at this point to fill the muscles in time, so the delay gets big. Other experiments showed that a maximum speed of approximately 0.3 $\frac{m}{s}$ results in a acceptable maximum deviation (e.g. in a turning point) of 4 cm.

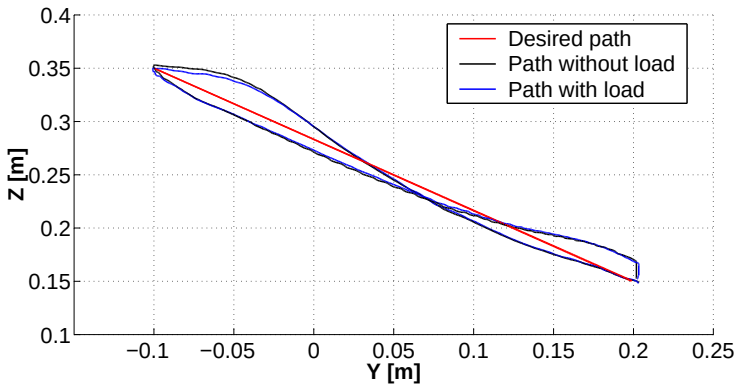

Figure 7.7: Reaching to point - front view

7.3 Control reaction on occurring disturbances

To judge the stability of the control against occurring disturbances, the arm was moved to a position, then pushed by hand, released quickly and the response joint responses were plotted in Figure 7.8. Only runs of the first three joints (kinematic numbering) were recorded here to keep the diagram clear. The other joints have to move less masses and are even faster in their response so that their response time is not of interest.

As can be seen, after releasing the arm at t=0.75 s , it takes another 0.75 s to reach the desired positions for all joints. Elbow and lifting shoulder joint have very little over shoots (6-10 %), the turing shoulder joint has an over shoot that is with 20 % larger, one reason for this problem is definitely the slackness in the mechanics of the first joint. An improvement here would definitely reduce the overshoot.

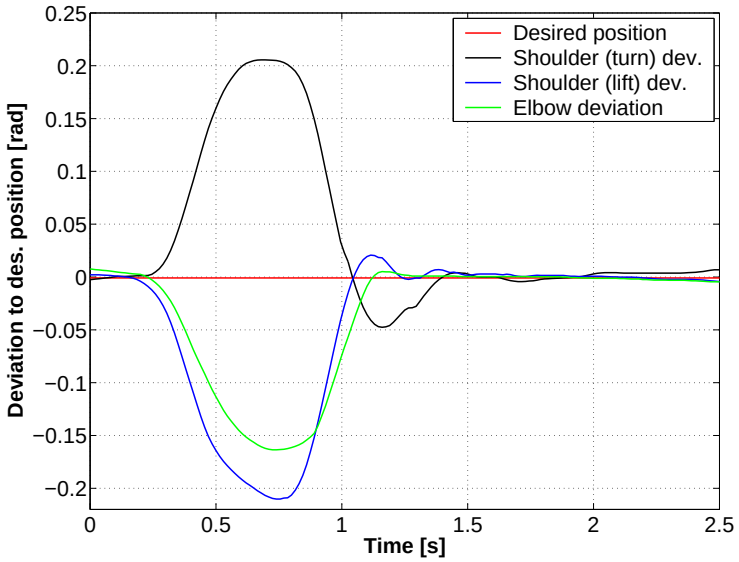

Figure 7.8: Reaction of joints 1-3 on occurring disturbances

Chapter 8

Conclusion and Outlook

8.1 Conclusion

In this work, the ability of a cascaded controller architecture to control artificial muscles was shown. The controller was tested for fast and exact following of a desired trajectory, stability against parameter changes and stability against occurring disturbances were proven.

It was shown that a compensation of static torques like gravity leads to much better control quality, but that also compensation of dynamic effects improves the control veritably.

The comparison with a regular PID controller on the testbed leads to the conclusion that higher control architectures can handle the control difficulties of artificial muscles better and are to be preferred.

Using an actuator model to predict the torque and to use it as a feedback for the torque control loop seems to be a good alternative to using a real torque sensor. The parameter identification of the torque model is time-consuming and needs to be carried out very carefully. Anyway, using an actuator model is cheaper and needs no effort to integrate and calibrate sensors, so that these disadvantages might be revised.

8.2 Outlook

The introduced control architecture can be implemented in any upper level control that is generating a desired trajectory. The trajectory can be applied in XYZ coordinates or by directly giving joint angles to the control. One possible application is the integration in a behavior-based control, where the movement is generated on the basis of primitive behaviors and the resulting trajectory is send to the control.

The following changes can be made to improve the control and the potential of ISAC:

- Extending the torque compensation part by adding more dynamic models
- Attaching real torque sensors to the joints
- Replacing the Shadow and Bridgestone muscles with Festo Fluidic Muscles

The compensation part of the control can be extended with more dynamic models for other joints. This would improve the movement especially under the circumstances of a higher arm velocity. Experiments whether the speed of the arm can be increased much should be carried out first to decide if it is useless to consider more dynamic effects.

Attaching real sensors to the arm would make the parameter identification of the actuator model unnecessary, improve the control quality especially concerning the influence of

parameter changes and disturbances. If this is worth the effort could be tested on just one joint to see the results.

The main problem in the present architecture is the usage of both Bridgestone and ISAC muscles. Combination of different types makes the parameter identification difficult and increases the complexity for the system. Due to the fact that Bridgestone Rubbertuators are not manufactured any more, it is only a question of time to replace them with Shadow Air Muscles. The Shadow muscles are not as strong as Bridgestone's, and might not be used to actuate joints with high load, like shoulder or elbow. Festo Fluidic Muscles are strong and feature less hysteresis because of their multi-layer architecture (Section 2.1.5). The entire arm could be provided with these muscles to have a homogeneous system. A proposal to change the Shadow and Bridgestone actuators to Festo Air Muscles is attached in Appendix F.

Appendix A

Abbreviations

CHR	:	Chien, Hrones and Reswick method, tuning method for PID controllers
CIS	:	Center for Intelligent Systems
DH	:	Denavit Hartenberg, convention to set joint frames
IAIM	:	Institute for Industrial Applications in Informatics and Microsystems
	:	University of Karlsruhe (TH)
IRL	:	Intelligent Robotics Laboratory at Vanderbilt University
ISAC	:	Intelligent Soft Arm Control
NI	:	National Instruments, manufacturer of industrial measurement devices
PID	:	Controller architecture with Proportional, Integral and Differential part
PI-PI	:	Regular cascaded controller architecture with Proportional, Integral controller for inner and outer loop
PI-MBPI	:	Model based cascaded PI-PI controller
PI-CMBPI	:	Model based cascaded PI-PI controller architecture with Compensation of static and dynamic parts
TCP	:	Tool Center Point, geometric grasp center of the robot hand
ZN	:	Ziegler-Nichols method, tuning method for PID controllers

Table A.1: Abbreviations

Appendix B

Elucidations

B.1 Table of symbols

α_i	:	Link parameter describing angle between Z_i and Z_{i+1}
Δp	:	Pressure difference between muscles and initial pressure p_0
Δp^i	:	Current, corresponding to pressure difference
ϵ	:	Contraction ratio
φ	:	Joint angle in internal joint position ($\pm\pi$ range)
φ_a	:	Actual joint angle
φ_e	:	Joint angle error
φ_d	:	Desired joint angle
τ_a	:	Actual torque caused by or acting on joint
τ_d	:	Desired torque caused by or acting on joint
τ_e	:	Torque error
Θ_i	:	Link parameter describing angle between X_{i-1} and X_i
ζ	:	Joint damping coefficient (testbed)
a_i	:	Link parameter describing distance from Z_i and Z_{i+1}
A, B, C, D	:	Actuator model parameters
d_i	:	Link parameter describing distance from X_{i-1} and X_i
d_{m_i}	:	Distance from joint axis one to mass i
D_0	:	Muscle diameter
F	:	Force acting on or caused by artificial muscle
F_0	:	Open loop transfer function
F_C	:	Transfer function of pressure control unit)
F_{C_τ}	:	Transfer function of torque controller (cascaded controller)
F_{P_τ}	:	Transfer function of position controller (cascaded controller)
F_J	:	Transfer function of joint (testbed)
F_{PID}	:	Transfer function of PID controller
I	:	Moment of inertia

Table B.1: Table of symbols (1)

K_C	:	Gain of pressure control unit
K_D	:	Differential gain PID controller
K_I	:	Integral gain PID controller
$K_{I,P}$	:	Integral gain position controller (cascaded controller)
$K_{I,T}$	:	Integral gain torque controller (cascaded controller)
K_J	:	Joint gain (testbed)
K_P	:	Proportional gain PID controller
$K_{P,P}$	:	Proportional gain position controller (cascaded controller)
$K_{P,T}$	:	Proportional gain torque controller (cascaded controller)
K_P	:	Proportional gain PID controller
$l_{1..5}$	:	Soft Arm link lengths
$m_{1..3}$	:	Compensation masses
p_0	:	Initial pressure
$p_{1,2}$	:	Pressure in muscles one and two
r	:	Chain wheel radius
T_C	:	Lag time of pressure control unit
T_J	:	Joint lag time (testbed)
${}_{i}^{i-1}T$	:	Transformation matrix describing transformation from link i-1 to link i

Table B.2: Table of symbols (2)

Appendix C

ISAC Hardware Specifications

C.1 Artificial muscles

C.1.1 Bridgestone Rubbertuators

One kind of muscles used in ISAC and testbed are Bridgestone Rubbertuators. The specifications of the different types are the following:

Rubbertuator Type	#5	#10	#14	#18
Tolerance Capacity	$0-5kg/cm^2$	$0-6kg/cm^2$	$0-6kg/cm^2$	$0-6kg/cm^2$
Contraction Tolerance	$0-20\%$ of eff. l.	$0-20\%$ of eff. l.	$0-20\%$ of eff. l.	$0-20\%$ of eff. l.
Operating Temperature	$-10-50°C$	$-10-50°C$	$-10-50°C$	$-10-50°C$
Operating (max)	85% RH	85% RH	85% RH	85% RH
Elasticity Strength	100 kg	250 kg	650 kg	750 kg
Effective Length	150 mm	200 mm	300 mm	400 mm

Table C.1: Rubbertuator specifications

C.1.2 Shadow Air Muscles

Due to the fact that Bridgestone Rubbertuators are no longer manufactured, Shadow Air Muscles [9] are used in some joints and will replace Rubbertuators in the future.
The maximum pressure for all Air Muscles is 6 bar, the pressure should not exceed 3 bar in unloaded condition.

C.2 Bridgestone pressure control unit

Three different servo valve units are produced by Bridgestone. The type number SVO102-06 is used in the testbed. The same unit is used to control the right arm, SVO102-05 is pressures the left arm. The ± 15 V supply for the servo valve unit comes from a separate power supply.

Air Muscle	#6	#20	#30
Diameter	6mm	20mm	30mm
Length (Fully Stretched)	150mm	210mm	290mm
Weight	10g	40g	80g
Pull (3.5bar)	3kg	12kg	35kg
Maximum Pull	7kg	20kg	70kg

Table C.2: Air Muscle specifications

Type	SVO102-05	SVO102-06
Input Signal	4-20 mA	4-20 mA
Monitor Output	1-5 V	1-5 V
Pressure Control Range	$0.5 - 4.5 kg/cm^2$	$0.5 - 5.0 kg/cm^2$
Pressure Control Precision	$\pm 1.0\% FS$	$\pm 1.0\% FS$
Air Supply Pressure	$5.0 kg/cm^2$	$6.0 kg/cm^2$
Power Supply Voltage	$+15V(0.5A)$ $-15V(0.1A)$	$+15V(0.5A)$ $-15V(0.1A)$
Pressure Tolerance	$8 kg/cm^2$	$8 kg/cm^2$
Weight	$1.1 kg$	$1.1 kg$
Input Impedance	250Ω	250Ω

Table C.3: Bridgestone pressure control unit specifications

C.2.1 Trimming the servo valve unit

The 'Zero' and 'Span' trimmers are used to adjust both channels. Once a calibration is needed, the procedure can be done in the following way:

- Turn on the power supply, make sure the air supply pressure is 6.0 kg/cm^2.
- Apply 4 mA current to the servo unit and read the resulting pressure with a external pressure indicator or with the monitor output of the servo valve unit. Use the 'Zero' trimmer to adjust the pressure to 0.0 kg/cm^2.
- Apply 20 mA current to the servo unit and use the 'Span' trimmer to make adjustments that the resulting pressure is 5.0 kg/cm^2.

C.3 Bridgestone encoder buffer

The encoder buffer transforms the 5V resp. 12V signal from the encoder to RS 232 level for use with longer distances. The encoder buffer acts also as power supply for the encoder itself, where a small switch changes between 5V or 12V encoder voltage. Three encoder phases are simply connected to A, B and Z at the encoder, whose outputs PHA, PHB and PHZ are connected to the PC Card. The encoder buffer itself is supplied also like the servo valve unit by the $\pm$ 15 V power supply.

Terminal No.	Specification
1	-15 V
2	0 V
3	+15 V
4	input signal (+)
5	input signal (-)
6	GND
7	Monitor output (+)
8	Monitor output (-)

Table C.4: Bridgestone pressure control unit pin assignment

C.4 PC-based Soft Arm controller card

The development of a Soft Arm controller card was the subject of Steve Northrup's master's thesis at the CIS. It is used to send analog signals corresponding to desired pressures to the valves and for reading as well as counting the encoder steps.

Pin No.	Specification	Pin No.	Specification
1	GND	20	GND
2	ENC1Z	21	ENC2Z
3	ENC3Z	22	ENC4Z
4	ENC5Z	23	ENC6Z
5	ENC1A	24	ENC1B
6	ENC2A	25	ENC2B
7	ENC3A	26	ENC3B
8	ENC4A	27	ENC4B
9	ENC5A	28	ENC5B
10	ENC6A	29	ENC6B
11	GND	30	GND
12	V1ch1	31	V1ch2
13	V2ch1	32	V2ch2
14	V3ch1	33	V3ch2
15	V4ch1	34	V4ch2
16	V5ch1	35	V5ch2
17	V6ch1	36	V6ch2
18	GND	37	GND
19	GND		

Table C.5: Soft Arm controller card pin assignment

Each card can drive 6 joints, which means 12 muscles and 6 encoders. The analog signals are of 4-20 mA standard and refer to a common mass. Table C.4 shows the pin assignment of the 37 pole SubD connector of the card.

Appendix D

ISAC Software

D.1 Files overview

`softarm.h`

Header file for the Soft Arm control.

`softarm.cpp`

Contains the main function, the control loop itself and all other functions.

`control.h`

Contains the definition of class *joint* and prototypes of membership and regular functions.

`control.cpp`

File that contains torque and position control algorithms, as well as computation of values for the compensation part and the PI-PI parameter sets.

`kinematics.mws`

Maple work sheet, containing all equations to compute direct and inverse kinematics. Transformation matrices, etc.

`statdyneffects.mws`

Maple work sheet. All formulas needed for gravity and moment of inertia compensation are included here.

`exp1.avi`

Video file of the experiment when the arm was following a triangle trajectory without load.

`exp2.avi`

Shows following a triangle trajectory with load.

`exp3.avi`

Video file of the experiment moving the arm straight towards destination point and back.

`exp4.avi`

Experiment when moving the arm in Z direction up and down a couple of times without load.

`exp5.avi`

Video file of same experiment then exp4.avi now with load.

`exp6.avi`

Shows video file moving the arm along a couple of points in space with load.

D.2 Description of C-functions

D.2.1 Membership functions

```
void InitJointPositions(void);
```

This function moves the arm slowly to the offset position. It is useful to check whether the variables *Offset_p* are still right. After executing this function, the joints should be in positions defined in *startpos*.

```
void PositionControlLoop(void);
```

This function contains the position controller algorithm. First, the integral error is computed by differentiating the desired position *r_p* minus the actual position *x_p* and is stored in *IntError_p*. The controller output is computed and stored as *u_p* before a check is performed whether the controller has already reached the limit which is within *LCLimit_p* and *UCLimit_p*. If this is the case, the error is not increased in this cycle, so the computed *IntError_p* value is subtracted again. This check is needed to avoid overflow of the controller error and to keep the control stable.

```
void ReadEncoder(void);
```

Reads low byte and high byte, computes the actual joint position and makes sure the position is between -π and π. Velocity and acceleration are computed and old values are stored.

```
double ReadEncoderInit(void);
```

Reads low byte and high byte of an encoder and returns the sum.

```
void SendOutput(short);
```

Sends two pressures to the servo valves by adding and subtracting the received value to the defined bias pressure. Before sending the value to the port, a check whether it is between 0 and 4095 is performed.

```
void SetPIDGain(double , double , double, double, double, double);
```

Sets the PID parameters according to the received values. Three values for each controller, position and torque, are set.

```
void SetJointParam (double, double, double, double, double);
```

Sets the joint parameters, which are four muscle parameters and the radius of the chain wheel.

```
void TorqueControlLoop(void);
```

This function is almost same than the function *PositionControlLoop*. The torque control algorithm is similar to the position control algorithm, different is the additional part d_T when computing the torque error and the conversion *u_T=Pressure2DA(u_T)*, which changes the unit of the controller output u_T $[kg/cm^2]$, to a corresponding integer value. Finally, the computed pressure difference value is send to the port.

D.2.2 Regular functions

```
void ArmToJointAngles(void);
```

This function is needed to convert joint angles, set according to the DH convention (these joint angles are called *arm angles*), into internal or real joint angles (called *joint angles*). The *arm angles* are used in kinematics, the joint angles in the controller algorithms and all muscle-related operations (joint limitations, actuator model). Each joint angle *joint[i].armangle* is hereby converted to *joint[i].jointangle*. Note that only joints 1 and 2 can be directly converted, joints 3-6 are combined as pairs 3-4 and 5-6.

```
void Calibration_x0(void);
```

This function is executed at the beginning to calibrate the arm. A high pressure is given to both muscles of each joint, so that the joint is very stiff and almost in the middle position where both muscles have same length. In this position, the actual encoder value is stored in *x0* and the joint position x_p is set zero.

Afterwards, muscle pressure is reduced to normal initial pressure which is *joint[i].BiasU*.

```
void CheckIfReachable(void);
```

This function checks the first column of *TrajectoryData* for a value of 3.0, which means that no valid configuration of joint angles exists for this XYZ position, so that the point is not reachable. The check itself is done in the *InvKinematics* function, this function displays only the result for the user by printing *reachable* or *NOT reachable* on the screen.

```
void DirKinematics(double theta1..double theta6);
```

The position of the TCP is computed in this function, according to given arm angles $\Theta_1 - \Theta_6$. If the actual joint angles want to be used in the direct kinematics, the conversion must be made with *JointToArmAngles* before calling this function. Depending on how the function is used, the XYZ position of the wrist or of the TCP can be done, according to the equations derived in 4.3.1. The resulting position is stored in *pos[0]*, *pos[1]* and *pos[2]*.

```
double getDist(double phi1, double phi2);
```

Returns the shortest distance of *phi2*, relative to *phi1* in the range $\pm\pi$.

```
void GetInterpolPoints(double DestX, double DestY, double DestZ);
```

This function computes interpolated points between the actual TCP position and the destination given in *double DestX, double DestY, double DestZ*. The euclidic distance of these points is set in the global variable *intpoldist*. The actual TCP position is found by reading the internal joint angles, converting them to kinematic angles and computing the direct kinematics.

Then, interpolated points are computed on a straight line between actual and desired position with the distance intpoldist. The X,Y,Z coordinates of the points are stored in the array *TrajectoryData[i][1]*, *TrajectoryData[i][2]* and *TrajectoryData[i][3]*. The first column in *TrajectoryData* contains a status value, this is set to 0.5 for each row where an interpolated point was stored.

```
void GetStatDynModel(void);
```

This function computes torque values caused by gravity and inertia of masses m_1, m_2, m_3. The inertia torque is considered for joint 1 only, the gravity torque for joints 2-6. Note that these joints are similar to those in Figure 4.3, which means that the torques resulting torques have to be transformed to real joints later.

First, distances of masses m_1, m_2, m_3 to axis 1 are determined to compute the masses of inertia later. Each mass of inertia causes a torque on the first joint, the sum is stored in *joint[0].CompTorque*.

Next, the torque caused by gravity and acting on joints 2-6 is computed. The computation for each joint is a summarization of the torques caused by masses 1,2,3. *Tau23* for example contains the torque, which is acting on joint 4 and caused by mass 2. The difference in joint numbering is because joint[3] is actually joint number 4.

The computed torques for each joint are summarized and stored in *joint[i].CompTorque*.

```
void InvKinematics(void);
```

This function is computing, if possible, a set of arm angles $\Theta_1 - \Theta_6$ out of the XYZ position in *TrajectoryData[i][1] - TrajectoryData[i][3]*, and storing them in the array *TrajectoryData[i][4]* til *TrajectoryData[i][9]*.

The computation of the angles follows the procedure presented in Section 4.3.2. After this computations, the arm angles have to be transformed into joint angles, to be able to check whether they are within the joint limits *joint[i].LLimit* and *joint[i].ULimit*. Now it is clear which of the four computed sets of angles is still valid. Out of these, the one which is closest to the actual arm position is chosen and stored in *TrajectoryData*. At the same time, *TrajectoryData[i][0]* is set 2.0, if there is no valid configuration at all, this value is 3.0.

```
void JointToArmAngles(void);
```

This function is almost same than *ArmToJointAngles(void)*, just working in the other way and used to convert the angles *joint[i].armangle* into *joint[i].jointangle*.

```
double NormAngle(double phi_in);
```

This functions exists in two versions as *NormAngle(double phi_in)* and *NormAnglePMPI(double phi_in)*. The difference is that *NormAngle(double phi_in)* returns an angle in radian, corresponding to the input in radian but in the range $0 - 2\pi$, where *NormAnglePMPI(double phi_in)* returns a value in radian within the range $\pm\pi$.

```
void SaveTrajectoryData(void);
```

This function stores the entire content of the array TrajectoryData in a file with the same name. The file looks exactly same like the array, values in a row are each separated by a blank.

```
void SoftArmInit(void);
```

This function is the first one to be called when starting the arm control. It initializes variables like gains, lower and upper joint limits, offsets, start positions, joint parameters and other values that need to be pre-defined and to be changed sometimes.

Appendix E

Testbed

In this chapter, components and software used in the testbed are described. Because some hardware is similar to ISAC's, please see Appendix C for information about Artificial Muscles, Bridgestone Servo Valve Unit, Bridgestone Encoder Buffer or PC-Based Soft Arm Controller Card.

E.1 Hardware

E.1.1 Force sensors

Two Futek L2353 force sensors are used to measure the force, given by each muscle and to compute the torque out of the difference according to $T = (F_2 - F_1)r$. The sensors are connected to amplifiers to increase the signal to 0-10 V. The capacity of the sensors is with 200 N max. high enough to cover the full operation range of the used muscles. Additional overload-stops protect the sensor against possible damage. Wiring code of the sensor connection is:

RED	Excitation+
BLACK	Excitation-
GREEN	Signal+
WHITE	Signal-

Table E.1: Force sensor pin assignment

E.1.2 Pressure sensors

Futek pressure sensors P4010 with a maximum capacity of 300 psi are used to measure pressure as close to the muscles as possible.

RED	Excitation+
BLACK	Excitation-
GREEN	Signal+
WHITE	Signal-

Table E.2: Pressure sensor pin assignment

E.1.3 Strain gauge amplifiers BA 501

Miniature amplifiers are used to amplify the strain gauge signal to a standard 0-10 V level. The following data sheet shows the specifications, the type for both force and pressure sensors is the BA 501. The voltage supply for the amplifiers is 18V and comes from a stabilized power supply.

Table E.1.3 shows the amplifier's pin assignment:

Pin No.	
1	Excitation-
2	Signal-
3	Signal+
4	Excitation+
5	Supply+
6	Com
7	Output

Table E.3: Strain gauge amplifier pin assignment

More information (to change sensitivity, if necessary) can be found at the web page [11].

Specification	BA 501
Output	0.1 to 10 V
Bridge Sensitivity	2.2 to 3 mV/V
Bridge Excitation	8 V
Bridge Resistance	350-5000 Ω
Bandwidth	1000 Hz
Nonlinearity	0.02 %
Operating Temperature	-40..85 $°C$
Dimensions	dia. 19.5 x 12 mm

Table E.4: Strain gauge amplifier specifications

E.1.4 National Instruments multi-IO card

The National Instruments multi-IO card LAB PC 1200 AI is used to read the amplified signals from force and pressure sensors (0-10 V), and can also read the monitor output of the servo valves. The analog input polarity is set to unipolar mode (0-10 V), the analog input mode is referenced single-ended (RSE).

The card preferences can be set with the national instruments software, called *Measurement and Automation Explorer*. This program also allows monitoring inputs as well as setting outputs to check the connection or to test a channel really quick. Card preferences can also be set with C-functions, see the software Section E.2.1 for that.

The following table contains the assignment of the card's 50 pole connector:

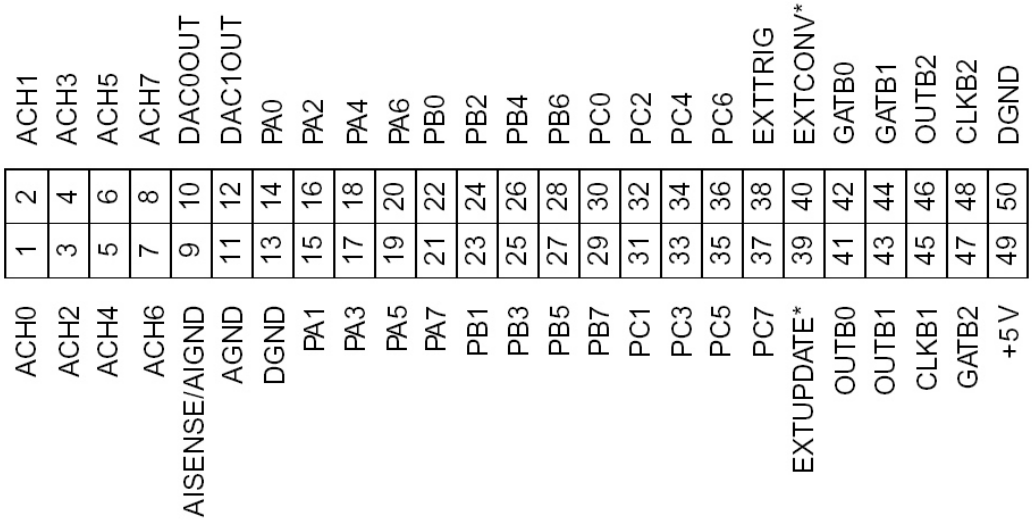

Figure E.1: Pin assignment of National Instruments IO card connector

E.1.5 Testbed interface

The interface from PC to the testbed is a 25 pole SubD connector. The assignment of the connector is the following:

E.2 Software

E.2.1 C-Functions for accessing the National Instruments card

The following header files have to be included to be able to use the library given by National Instruments which contains several C-functions to access the hardware:

```
#include"nidaq.h"
#include"nidaqex.h"
```

A list of all functions can be found in the *Measurement and Automation Explorer*, but notice that not all functions are available for the LAB PC 1200 AI. To read analog sensor signals, only two functions need to be used.

```
status = AI_Configure(deviceNumber, chan, inputMode, inputRange,
polarity, driveAIS);
status = AI_Read(deviceNumber, chan, gain, reading);
```

The function *AI_Configure* sets the card parameters, which have the following meaning:
In our case, the function is called as *AI_Configure(1, -1, 0, 0, 1, 0);*. The function *AI_Read* is used to read and amplify the analog signal, the result is in 0 to 4095 range for the LAB PC 1200 in unipolar mode. Parameter specifications are shown in Table E.2.1.
Without amplifiers, the highest gain (100) was selected. When later amplifiers were mounted on testbed, the normal 0-10 V range was used and the gain was set to one. A typical call

Pin No.	Specification Testbed Side	Specification NI Card/Supply Side
1	V+ supply	+18V Power supply
2	GND supply	GND Power supply
3	n.c.	n.c.
4	n.c.	n.c.
5	FS1 Signal +	ACH 0
6	FS1 GND	AGND
7	FS2 Signal +	ACH 2
8	FS2 GND	AGND
9	PS1 Signal +	ACH 4
10	PS1 GND	AGND
11	PS2 Signal +	ACH 6
12	PS2 GND	AGND
13-25	n.c.	n.c.

Table E.5: Testbed interface pin assignment

Name	Description	Value
deviceNumber	assigned by Measurement and Automation Explorer	1
chan	channel to be configured	0-7, -1 for all channels
inputMode	single-ended or differential input use	0: differential 1: Referenced Single-Ended 2: Nonreferenced Single-Ended
inputRange	voltage range of analog	ignored
polarity	unipolar or bipolar configuration	0: Bipolar 1: Unipolar
driveAIS	ignored for this device	always set to zero

Table E.6: AI_Configure function parameters

for this function to read data from a sensor connected to the fifth channel which gives 0-5 V output could be *AI_Read(1, 4, 2, Value);*.

E.2.2 C-functions for controlling the testbed

Several basic functions can be used to read sensory data, to send desired values to the testbed and can be implemented in higher level functions.

```
void getEncoderData();
void getSensorData();
void SendDataToValves(USHORT p1, USHORT p2);
void SendDeltaPressure(DOUBLE p),
int EncoderData(LONG *EncoderValue);
```

The file example.cpp shows a sample of how to use these functions. It writes time, computed torque according to the actuator model and other sensor information into a file. Due to the

Name	Description	Value
deviceNumber	assigned by Measurement and Automation Explorer	1
chan	analog input channel number	0-7
gain	gain setting for the channel	1, 2, 5, 10, 20, 50, 100
reading	result of the conversion	0 to 4095

Table E.7: AI_Read function parameters

non-real time ability of windows, the experiments have to be carried out with attention concerning simultaneous running of different tasks. During measurements, no other applications should be open or in use.

Description of functions

```
int EncoderData(LONG *EncoderValue);
```

This function is the regular function used in IMA to read encoder values, except that this one returns always value of encoder number 0, because only this one is connected when using the testbed. The value is a number of steps relative to the encoder zero position. The encoder which is used in the testbed has 8000 steps / revolution.

```
void getEncoderData();
```

Once the encoder value was read with the function above, this one computes the joint angle in radian according to a middle position of the joint. For this reason, the encoder value in middle position is stored in *EncMiddlePosition*. *CalEncoder* contains the number of steps for one turn (2π). At the same time, the joint speed is computed by deriving the joint angle difference within one time cycle.

```
void getSensorData();
```

This function reads force and pressure sensor information. The values of the different channels are summarized for a period of time (for 50 ms in example.cpp), to filter the signals digitally. This was necessary before connecting the amplifiers, because the noise was high reading the strain gauges directly so that this method gave better results. (the gain was adjusted to 100 in this case) After summarizing the values, the average was taken to compute physical values. The -Max and -Offset values have to be set in calibration, the computing is later just a linear transformation.

```
void SendDataToValves(USHORT p1, USHORT p2);
```

The same function used in IMA is also taken here to sent desired pressure values to the pressure control unit. The only difference is that all outputs are set to zero, except the two first ones where both muscles are connected. The input values for this function have to be in the range of 0 to 4095.

```
void SendDeltaPressure(DOUBLE p),
```

This function receives the desired delta pressure for the muscles as input. The unit of this value is kg/cm^2. The pressures are set according to $p_1 = p_{01} - \Delta p$ and $p_2 = p_{02} + \Delta p$. First, the values for both init pressures are computed. Both muscles have the same physical initial pressure, but different values might have to be send to the port to reach this pressure. Second, the converted delta pressure is added and both values are send to the port.

E.3 Calibration

To convert the values of the A/D port to physical values, some variables have to be set in a calibration process:

```
#define CalPressure1
#define CalPressure2
#define CalP1Offset
#define CalP1Max
#define CalP2Offset
#define CalP2Max
#define CalF1Offset
#define CalF1Max
#define CalF2Offset
#define CalF2Max
```

E.3.1 Calibrating the pressure control unit

To make sure the pressure control unit is giving the desired pressure, corresponding to the current input signal, the variables *CalPressure1* and *CalPressure2* have to be set. By giving the highest output of 4095 to both ports, the resulting pressure can be read from the mechanical pressure sensor or with the calibrated miniature pressure sensors. The zero trimmer has to be set correctly, because there is no offset designated in the program. The resulting value should be about 5.0 kg/cm^2 if both trimmers have been set correctly.

E.3.2 Calibrating pressure sensors

The pressure offsets *CalP1Offset* and *CalP2Offset* are the values read from the ports with a pressure of 0.0 kg/cm^2. To determine the values corresponding to $5.0kg/cm^2$, the best way is to connect both sensors directly to the pressure supply, and to adjust the pressure regulator to 5.0 kg/cm^2. The offset has to be subtracted from the values read from ports and the results are stored in *CalP1Max* and *CalP2Max*.

E.3.3 Force sensor calibration

To find the values for *CalF1Offset* and *CalF2Offset*, the sensors are placed flat on a board with no force acting on the sensor. The value read in this state is stored. A specific weight is now attached to the force sensor in vertical way. In our case, 4 gallons of water have been used to apply 149.1 N to the sensor. From the measured value, the offset has to be subtracted and the result is stored in *CalF1Max* and *CalF2Max*.

Appendix F

Festo Fluidic Muscles

To make a proposal how to change actual Shadow or Bridgestone muscles to Festo muscles, the specifications of Festo Fluidic Muscles are listed in Table F.

Muscle Type	MAS-10	MAS-20	MAS-40
max. Pressure	8 bar	6 bar	6 bar
Diameter	10 mm	20 mm	40 mm
Length min.	40 mm	60 mm	90 mm
max.	4500 mm	4500 mm	4500 mm
max. Force	400 N	1200 N	4000 N
Contraction	20 % Length	20 % Length	25 % Length
Hysteresis	< 5 %	< 5 %	< 5 %
Weight for min. Length	0.086 kg	0.28 kg	0.87 kg

Table F.1: Specifications of Festo Fluidic Muscles

Table F shows actual muscles used in different ISAC joints and proposes corresponding Festo muscles to replace them.

Joint Number	Muscle Length [mm]	Bridgestone Rubbertuator	Shadow Air Muscle	Festo Fluidic Muscle
1	450	#18, 245S	not available	MAS 40
2	300	#14, 830V	#30, 290mm?	MAS 40
3,4	150	#10, 415S	#20, 210mm	MAS 20
5,6	120	#5, 012S	#20, 210mm	MAS 10

Table F.2: Actual muscle types in ISAC joints and similar Festo Fluidic Muscles

For the first joint is no Shadow muscle available. The 290 mm Shadow muscle might not be able to lift the shoulder joint, according to the specifications of this muscle and the corresponding Bridgestone Rubbertuator. A complete replacement of Bridgestone Rubbertuators with Shadow Air Muscles is therefore not possible and it has to be reverted to Festo Muscles.

Appendix G

Transformation Matrices

Regular matrices for direct and inverse kinematics:

$$ {}^{0}_{1}T = \begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c2 & -s2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s2 & -c2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c3 & -s3 & 0 & -l_3 \\ s3 & c3 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} c4 & -s4 & 0 & 0 \\ 0 & 0 & -1 & -l_4 \\ s4 & c4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c5 & -s5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s5 & -c5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{6}T = \begin{bmatrix} c6 & -s6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s6 & c6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{m_1}T = \begin{bmatrix} 1 & 0 & 0 & -0.5\ l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{m_2}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.5\ l_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{6}_{m_3}T = \begin{bmatrix} 1 & 0 & 0 & -l_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Special transformation matrices used in inverse kinematics:

$$ {}^{0}_{3}T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & -c_1c_2l_3 - s_1l_2 \\ s_1c_{23} & -s_1s_{23} & c_1 & -s_1c_2l_3 + c_1l_2 \\ -s_{23} & -c_{23} & 0 & s_2l_3 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{6}T = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & 0 \\ s_5c_6 & -s_5s_6 & -c_5 & -l_4 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & -(s_4c_5c_6 + c_4s_6)l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{4}T = \begin{bmatrix} c_1c_{23}c_4 - s_1s_4 & -c_1c_{23}s_4 - s_1c_4 & c_1s_{23} & c_1s_{23}l_4 - c_1c_2l_3 - s_1l_2 \\ s_1c_{23}c_4 + c_1s_4 & -s_1c_{23}c_4 + c_1c_4 & s_1s_{23} & s_1s_{23}l_4 - s_1c_2l_3 + c_1l_2 \\ -s_{23}c_4 & s_{23}s_4 & c_{23} & c_{23}l_4 + s_2l_3 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{6}T = \begin{bmatrix} c_5c_6 & -c_5s_6 & s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ -s_5c_6 & s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$
{}^{0}_{5}T = \begin{bmatrix} (c_1c_{23}c_4 - s_1s_4)c_5 & -c_1c_{23}s_4 - s_1c_4 & c_1s_{23} & c_1s_{23}l_4 - c_1c_2l_3 - s_1l_2 \\ s_1c_{23}c_4 + c_1s_4 & -s_1c_{23}c_4 + c_1c_4 & s_1s_{23} & s_1s_{23}l_4 - s_1c_2l_3 + c_1l_2 \\ -s_{23}c_4 & s_{23}s_4 & c_{23} & c_{23}l_4 + s_2l_3 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Transformation matrices used to compute mass distances from joint axis one:

$$
{}^{1}_{m_1}T = \begin{bmatrix} c_2 & -s_2 & 0 & -0.5c_2l_3 \\ 0 & 0 & 1 & l_2 \\ -s_2 & -c_2 & 0 & 0.5s_2l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{1}_{m_2}T = \begin{bmatrix} c_{23} & -s_{23} & 0 & -0.5s_{23}l_4 - c_2l_3 \\ 0 & 0 & 1 & l_2 \\ -s_{23} & -c_{23} & 0 & 0.5c_{23}l_4 + s_2l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{1}_{m_3}T = \begin{bmatrix} p_5c_6 - c_{23}s_4s_6 & -p_5s_6 - c_{23}s_4c_6 & c_{23}c_4c_5 + s_{23}c_5 & -(p_5c_6 - c_{23}s_4s_6)l_5 + s_{23}l_4 - c_2l_3 \\ s_4c_5s_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & -(s_4c_5c_6 + c_4s_6)l_5 + l_2 \\ p_6c_6 + s_{23}s_4s_6 & -p_6s_6 + s_{23}s_4s_6 & -s_{23}c_4s_5 + c_{23}c_5 & -(p_6c_6 + s_{23}s_4s_6)l_5 + c_{23}l_4 + s_2l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The substitutions

$$p_5 = c_{23}c_4c_5 - s_{23}s_5$$

$$p_6 = -s_{23}c_4c_5 - c_{23}s_5$$

are used in the last matrix.

Bibliography

[1] Dawei Cai and Hiroo Yamaura. A robust controller for manipulator driven by artificial muscle actuator. *IEEE Conference on Control Applications*, pages 540–545, September 1996.

[2] D. G. Caldwell, A. Razak, and M. J. Goodwin. Braided pneumatic muscle actuators. *IFAC Conference on Int. Autonomous Vehicles, Southampton, UK*, April 1993.

[3] Darwin G. Caldwell, Gustavo A. Medrano-Cerda, and Mike Goodwin. Control of pneumatic muscle actuators. *IEEE Control Systems*, pages 40–47, February 1995.

[4] Pablo Carbonell, Zhong-Ping Jiang, and Daniel .W. Repperger. A fuzzy backstepping controller for a pneumatic muscle actuator system. *International Symposium on Intelligent Control*, pages 353–358, 2001.

[5] Ching-Ping Chou and Blake Hannaford. Static and dynamic characteristics of mckibben pneumatic artificial muscles. *International Conference on Robotics and Automation*, 1:281–286, 1994.

[6] Ching-Ping Chou and Blake Hannaford. Measurement and modeling of mckibben pneumatic artificial muscles. *IEEE Transactions on Robotics and Automation*, 12(1):90–102, February 1996.

[7] Bridgestone Company. Rubbertuators and applications for robotics, 1986. Technical Guide No. 1.

[8] Futek Advanced Sensor Company. Homepage: http://www.futek.com.

[9] Shadow Robot Company. Homepage: http://www.shadow.org.uk.

[10] J. J. Craig. *Introduction to Robotics*. Addison-Wesley, 1998.

[11] BLH Sensors GmbH. Homepage: http://www.blh.de.

[12] T. Hesselroth, K. Sarkar, P. van der Smagt, and K. Schulten. Neural network control of a pneumatic robot arm. *IEEE Trans. Syst. Man., Cybernetics*, 24:28–38, 1994.

[13] H.F. Schulte Jr. The characteristics of the mckibben artificial muscle. *The Application of External Power in Prosthetics and Orthotics*, 1961. National Acamy of Sciences - National Research Council, Washington D.C.

[14] K. Kawamura, R. A. Peters II, S. Bagchi, and M. Bishay. Intelligent robotic systems in service of the disabled. *IEEE Transactions on Rehabilitation Engineering*, 3(1):14–21, March 1995.

[15] Thilo Kerscher. Development of a control for a walking machine driven by pneumatic muscles. Diploma thesis, Institute of Information Processing Technics, University of Karlsruhe (TH), August 2002.

[16] L. Merz and H. Jaschek. *Grundkurs der Regelungstechnik*. Oldenbourg Verlag, 1996.

[17] S. Northrup, N. Sarkar, and K. Kawamura. Biologically inspired control architecture for a humanoid robot. *IROS 2001, Maui, Hawaii*, 2001.

[18] Steve Northrup. A pc-based controller for the soft arm robot. Master's thesis, Vanderbilt University, Center for Intelligent Systems, Nashville, Tennessee, March 1997.

[19] University of Karlsruhe (TH). Homepage: http://www.uni-karlsruhe.de.

[20] University of Karlsruhe (TH) Research Center for Information Technologies. Homepage: http://wwwneu.fzi.de/ids/eng/.

[21] N. Sarkar, S. Thongchai, M. Goldfarb, and K. Kawamura. A frequency modeling method of rubbertuators for control application in an ima framework. *Proceedings of American Control Conference, Arlington, Virginia*, pages 1710–1714, June 2001.

[22] N. Tsagarakis and Darwin G. Caldwell. Improved modelling and assessment of pneumatic muscle actuators. *IEEE Conference on Robotics and Automation*, pages 3641–3646, April 2000.

[23] P. van der Smagt, F. Groen, and K. Schulten. Analysis and control of a rubbertuator arm. *Biological Cybernetics, Springer-Verlag*, 75:433–440, 1996.