Master's Thesis
MSC-059

Mechanik und Meerestechnik

Dynamic Model Inversion of a Soft Robot

Dynamische Modellinversion eines Softroboters

by
Sean Maroofi

Supervisors: Prof. Dr.-Ing. R. Seifried
Prof. Dr.-Ing. C. Cyron
Dr.-Ing S. Drücker
M. Grube, M.Sc.

Hamburg University of Technology (TUHH)
Institute of Mechanics and Ocean Engineering
Prof. Dr.-Ing. R. Seifried

Hamburg, September 2023

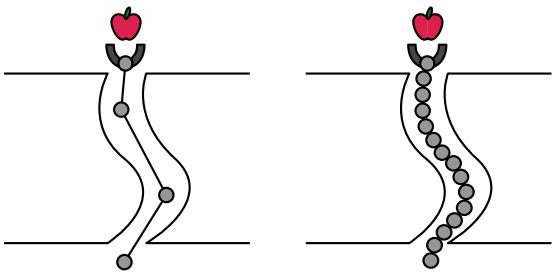
TUHH

Contents

Chapter 1

Introduction

Robotic systems have changed our everyday life over the last decade, not only performing dangerous tasks, as in the manufacturing of cars, but also incidental work in our everyday life, such as law mowing or vacuuming. Intensive research led to an increase in variety of robotic design, accurately constructed for their specific application. Most commonly, robots are manufactured out of rigid synthetic materials or metal. However, in some applications a rigid body foundation may be disadvantageous. As an example, Fig. 1.1 shows a simple scenario, where a robot arm should grab an apple in a very narrow environment. A rigid robotic arm would have to be constructed precisely for this task and may not be usable for another narrow surrounding. Instead, a flexible robot capable of forming an arbitrary shape would be ideal for such a task. These robots belong to the subcategory of soft robotics and are part of the continuum robots research field. Deformable robots are applicable in a variety of areas, such as medical procedures or underwater tasks . Their elastic nature allows easier interaction with patients, ensuring safety to their surroundings and the receiving end, as well as execution of careful tasks, such as grasping of fragile and sensitive objects. Examples for soft robots are shown in Fig. 1.2.



(a) Rigid robot. (b) Soft robot.

Figure 1.1: Rigid robots vs. soft robots in a simple example.

Soft robots come with a lot of benefits, but make the application of known general algorithms and mathematical concepts, which have been applied to rigid robots, more difficult at the same time. Conventional robotic arms consist of a series of rigid links connected by joints. Each joint can be controlled directly to reach desired locations in space. Contrarily, many soft robots are characterized by a continuous and jointless flexible backbone, which gives them the ability to take a variety of shapes [AmouriMahfoudiZaatri19]. Their elastic nature allows them to deform at any point along their bodies. The freedom in deformation causes an increase in the number of degrees of freedom, resulting in high kinematic redundancy. Consequently, non-linear structures and complex geometries hinder the derivation of analytical models to describe and control their motion [DingEtAl22].

Most of the existing control algorithms for soft robots rely on purely kinematic modeling [WebsterJones10], [ThuruthelEtAl18]. These static models are capable of controlling soft robots successfully. But to perform dynamic motion in space, pure kinematic modeling is not sufficient and mathematical models need to consider the dynamic behavior of soft robots. The development of real-time dynamic control strategies is difficult due to several reasons. As in the case of pure kinematic modeling, dynamic models have to take the infinite dimensionality of the robot's state space into account [Della SantinaEtAl20]. This leads to strong non-linear relations already for pure kinematic modeling. Consideration of dynamic relationships results in even more complex models. For example, distribution of differential masses, which affect the movement and rotational acceleration of the body, and differentiation of time dependent variables, which are already deeply nested and connected in the kinematic equations,

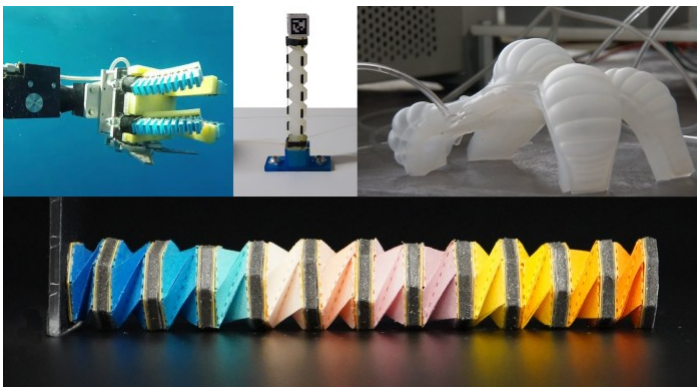


Figure 1.2: Examples for soft robots: Underwater soft gripper by [Wood16] (left), previous prototype from [Bekman22] (middle), NASA robot by [SullivanFlitzpatrick19] (right) and origami robotic arm by [WuEtAl21] (bottom).

complicate the mathematical derivation of a dynamic model [JensenEtAl22]. Large and complex differential equations emerge, which can be difficult and computationally expensive to solve, and for real-time implementations these models need to be designed in numerically stable form [WebsterJones10].

When it comes to deriving a forward model (determination of the robot's state given a system input) many works exist for representation of the dynamics [JensenEtAl22], [AmouriMahfoudiZaatri19], [ThuruthelRendaIida20]. But in order to perform output tracking tasks, the input to reach a desired output needs be determined. This requires inversion of the forward model, which is difficult due to its high non-linearity and complexity. Furthermore, soft robots are controlled with a limited amount of actuators and are consequently highly underactuated (infinite degrees of freedom). Therefore, not all degrees of freedom can be controlled independently, which further hampers the inversion [Drücker22]. But different approaches in literature exist to treat the inverse problem. One method is the framework of servo-constraints, which is suitable for complex underactuated multibody systems, and can be considered to solve the inverse problem [Kirgetov67].

The Institute of Mechanics and Ocean Engineering (MuM) owns a soft robot, which was manufactured and studied in previous works. After studies on kinematics and control of soft robots in [Wiek21] and [Wiek22], trajectory following was studied in [Bekman22]. As an extension to the previous work, this thesis is dedicated to the inverse dynamics problem for another self-constructed soft robot of bigger size, see Fig. 1.3. With the goal of trajectory following, a representation for the inverse model is derived using the servo-constraints approach. Following a short introduction, an overview of common modeling techniques for soft robots is presented in Chap. 2. A forward model is derived in Chap. 3. Afterwards, the inverse direction is addressed in Chap. 4 by applying the servo-constraints approach. Chap. 5 gives an insight into the implementation and provides details about the real-system. Chap. 6 is dedicated to evaluation of experimental results. Finally, Chap. 7 summarizes this work and presents an outlook on improvements and future work.

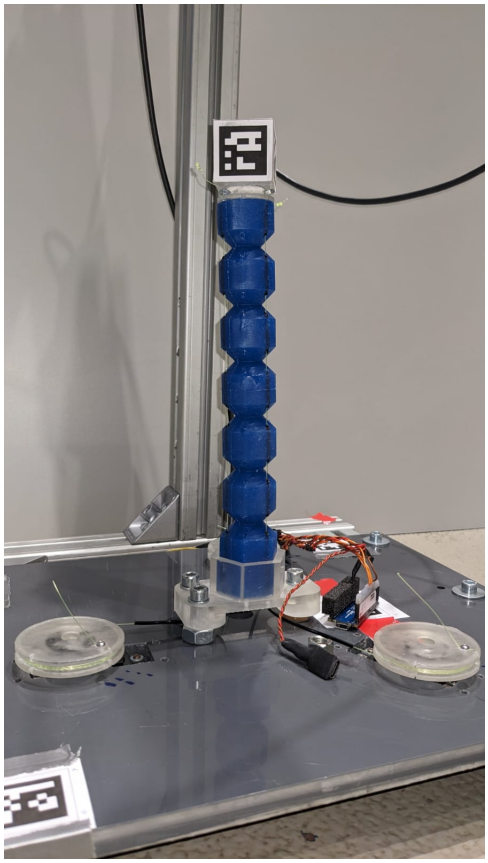


Figure 1.3: The Soft robot studied in this work.

Chapter 2

Fundamentals

This chapter provides an overview on the fundamentals of robotic modeling. First, the concept of operation spaces is introduced in Sect. 2.1, which gives an overview on the structure of traditional rigid robots. Afterwards, popular approaches in literature for modeling the forward mapping of soft robots are presented in Sect. 2.2. Finally, Sect. 2.3 summarizes present work on solving the inverse direction for soft robots and introduces other common approaches for derivation of inverse models for underactuated multibody systems.

2.1 Operation Spaces

Commonly, to model conventional rigid robots, the overall system is separated into different operation spaces [SicilianoKhatib08]. This becomes advantageous for defining relationships and functions for control of each sub-part. These can be considered separately and individual mappings in between these operation spaces can be established. Every operation space can be described by its own parameters in a chosen state, while the mapping functions provide the physical and mathematical relationships between these states.

Commonly defined spaces are the configuration space, the task space and the workspace [Mueller19]. The configuration space gives a specification of the position of all points of a robot in space. In terms of rigid robots, this is usually achieved by providing information about the current angle of revolute joints or present length of prismatic joints. Soft robots lack physical joints and need to be characterized with other parameters, which can describe the deformed state. An example for these are the curvature of beam shaped elements or current lengths of stretchable soft links. The task space describes the positions and orientations

of the end-effector at the tip of a robotic arm in space. Further, the workspace is defined as a subset of the task space, which contains the physically reachable positions and orientations of the end-effector.

In addition to the already mentioned spaces another space can be defined, which includes information about the current state of actuators. In terms of rigid robots it consists of the forces and torques applied to the joints by actuators, for example electric motors. Soft robots exist in a variety of forms and configurations, which is why the actuation methods vary from one to another. For example, pneumatic robots are actuated by changing the volume or present pressure of chambers, while tendon-driven robots possess electrical motors like rigid robots, which provide forces and torques again. [ThuruthelEtAl18] separate the actuation further by introducing an additional joint space. Commonly, the actuating components of the system are connected with intermediate tools to the actual soft robot. For example, tendon-driven robots do not utilize motors, which directly change the robots shape, but instead have cables attached to their body, which are pulled to change the robots configuration. Therefore, the joint space incorporates information about the current length of the attached cables of tendon-driven robots.

In order to formalize a complete system, all the spaces are connected with mapping functions, resulting in a forward direction and inverse direction. The overall system separation, including all spaces and mapping functions, is visualized in Fig. 2.1. The actuation space is connected to the joint space via a physical relationship, on how the lengths and resistance change once the actuators apply an input. This actuation mapping is dependent on the chosen manipulator and actuation system [WebsterJones10]. The relationship between joint space and configuration space is represented by the equations of motion of the system. Finally, the forward kinematics transform the state of the soft robot to task space coordinates, such as the position of the end-effector in space. For the inverse direction a mapping has to exist, which gives the requisite state given desired task space coordinates. The inverse dynamics take the mapping from necessary state configuration to needed inputs in the joint space. Lastly, an inverse mapping of the actuation relationship provides the required input.

In contrast to classic fully actuated rigid robots, a clear separation of the system into the depicted sub-moduldes in Fig. 2.1 is not straightforward for soft robots, especially in the case for the inverse direction. Rigid robots can be fully modeled by a finite discrete set of frames, while soft robots are a set of continuous particles [ArmaniniEtAl22]. Therefore, describing the soft robots behavior is challenging. Several solutions can exist for a suitable configuration to reach a point in space,

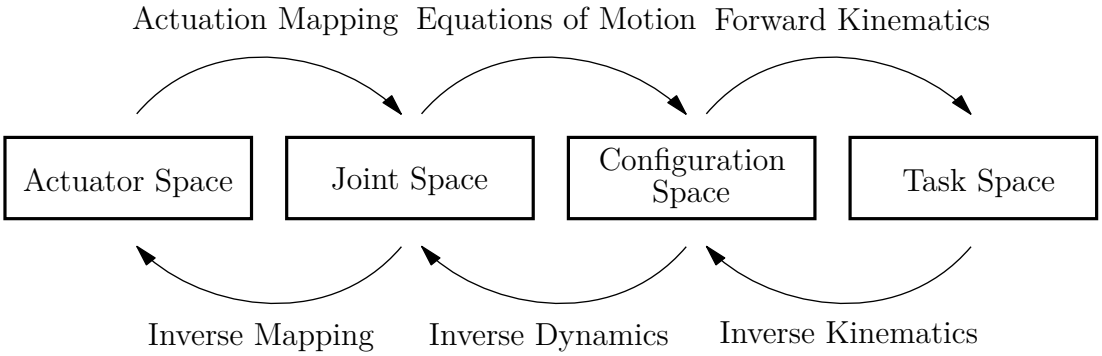


Figure 2.1: Separation into the different operation spaces.

making the forward mapping a non-injective function, potentially. Surjection is not guaranteed either. It can be the case, that not every desired outcome is possible to achieve, especially in the case of dynamic motion. Finding a fitting configuration for reaching a desired location with the end-effector can be troublesome for the inverse function. Therefore, bijection in general is not guaranteed for the forward and inverse transformation. These hurdles complicate the modeling of soft robots and do not always support a strict separation of the workspace. Some works exist, trying to describe each sub-mapping individually, others define a more compact model, incorporating multiple of the sub functions. Because of that, a separate consideration of forward and inverse direction is presented in the following. Some of the more common methods for modeling soft robots are introduced and explained. In particular for the inverse dynamics, an overview of available research is provided and an additional insight into common approaches applied to general underactuated multibody systems is presented.

2.2 Forward Direction

The forward direction aims to determine the system's output given a known system input. In that case, the actuation mapping is the first part of the forward chain. It depends on the actuation choice of the real system, and therefore its representation is rather straightforward and different for each individual system. In contrast, the modeling of the kinematics and dynamics depends on the requirements, which the soft robot model has to meet. The model assumptions describe the trade-off between accuracy and computational efficiency [ArmaniniEtAl22]. For example, online control or parameter optimization processes motivate quick computation time, while high accuracy is favorable for evaluation of new actuators and determination of structural loads.

2.2.1 Full Finite Element Model

To achieve accurate modeling of the deformation of a soft robot, finite element methods (FEM) can be considered. FEM is one of the most popular numerical calculation methods for solving continuum mechanic problems [ArmaniniEtAl22]. It allows incorporation of prior obtained knowledge of material properties and provides precise and effective modeling of deformations of flexible bodies [PolygerinosEtAl15]. FEM can be adapted to different complex geometries and can provide a deeper insight into internal properties, for example the interactions between layers of different materials [DingEtAl22]. Arbitrary geometry shapes can be analyzed which is especially compelling for soft robots with extraordinary shape [GouryDuriez18]. However, simulation by applying FEM showed to result in high computation time to represent the deformation of soft robots [CoevoetEtAl17], as deformations of cross sections are included during the computation process. Especially in the case of complex geometrical shapes a high number of voxels for modeling is necessary [DingEtAl22]. In that regard, tasks requiring fast computation can only be performed with further approximations, such as linearization or model reduction [ZhengLin22]. Researchers introduced workarounds, such as an asynchronous framework, that achieves a trade-off between accuracy and computation time [LargilliereEtAl15]. Other approaches exist which combine of FEM with piecewise constant curvature (PCC)-modeling (introduced in Sect. 2.2.3) [RungeEtAl17] to overcome the issue of high computational cost. In general FEM methods produce highly accurate models of soft robots, but are rather unsuitable for online real-time application.

2.2.2 Cosserat Rod

Many soft robots have a significant greater length compared to their cross section which motivates the strategy to model it as a flexible beam. The continuous Cosserat rod theory describes a long and thin soft robot by an infinite degrees of freedom model, without explicitly considering its volume [SpillmannTeschner07]. This is achieved by stacking an infinite number of infinitesimal cross-sections. In contrast to full FEM models, Cosserat rod models do not consider deformation of the cross-section, which results in more computation efficiency. The rod is characterized by its centerline along its longitudinal axis, depicted in Fig. 2.2. With the assumption of rigid cross-sections along the centerline a continuous function $\boldsymbol{g}(s,t)$ is defined, which describes the position and orientation of each cross-section with respect to a global frame [XunZhengKruszewski23]. Solving these continuous models is widely considered as overly complex and too costly for fast real-time tasks [TillAloiRucker19]. The motion and orientation for each point on the rod has to be calculated, which is why a discretization of a con-

tinuum Cosserat rod is necessary. This can either happen on numerical resolution level, when solving the continuous system directly, or by discretization of the Cosserat rod model [ArmaniniEtAl22]. The latter method of beforehand discretization of a continuous Cosserat rod allows faster computation times. For example, [TrivediLotfiRahn08] and [RendaEtAl14] model manipulators using continuous Cosserat rod theory, which produces highly accurate results but at the cost of computational complexity. Another cosserat rod model is studied in [RendaEtAl18] and shows better performance in terms of computation time.

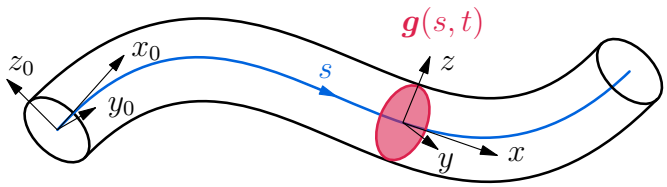


Figure 2.2: Cosserat rod parameterization by [XunZhengKruszewski23].

2.2.3 Piecewise Constant Curvature

One of the most popular approaches in modeling kinematics of a beam-like soft robots is the idea of constant curvature. The PCC-approach is a simple method for modeling the shape of soft robots. In comparison to the Cosserat rod model, the geometrical description is achieved with a smaller number of parameters [ShamilyanEtAl23], for example by the arc's curvature κ, the arc's length ℓ and rotational quantities, such as an angle θ describing plane rotation. Parameterization with these variables is depicted in Fig. 2.3. Methods, which are based on the assumption of constant curvature along a soft robots longitudinal axis, date back to [WebsterJones10]. This concept has been advanced to a discretization of a slender soft robot into multiple segments, as with greater length robots tend to show strong non-constant curvature. PCC has been widely studied in the soft robotics community. [FalkenhahnEtAl14] use the parameterization of [WebsterJones10] and derive a dynamic model following the Euler-Lagrange formalism. [RoneBen-Tzvi14] study forward dynamics, but with a different parameterization considering curvature β and γ in two direction in space, see Fig. 3.2. With PCC, computation time can be kept very low for a small amount of segments, while still providing accurate results. However, singularity issues have been reported in different studies [AllenEtAl20], usually appearing when the soft robot is in a straight configuration. Zero-division operations can lead to singularities not only in the kinematics but become more troublesome in the derivation of the system dynamics [JensenEtAl22]. Furthermore, some continuum manipulators can not be modeled precisely enough using PCC, such as conically shaped robot arms [RungeEtAl17], requiring other methods for kinematic modeling.

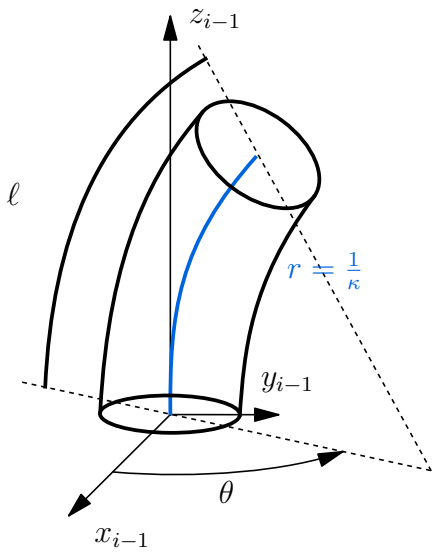


Figure 2.3: Constant curvature parameterization by [WebsterJones10].

2.2.4 Data-Driven Methods

Another option for precise representation can be achieved by applying data-driven models. Learning a model is a beneficial alternative to the aforementioned approaches, when low computation cost is desired and a simplification to a beam model is not possible. They come with the advantage of not requiring derivation of a physical model and provide fast computation, making them suitable for real-time control [ArmaniniEtAl22]. Furthermore, data-driven methods are capable of considering manufacturing errors by learning the inaccuracies, which arise during assembly or production of the soft robotic system. Instead of deriving complex analytical models and performing parameter estimation experiments, data-driven approaches require predetermined training data for approximating the highly non-linear functions of a soft robotic system. In that regard, [ThuruthelRendaIida20] design recurrent neural networks to approximate the forward dynamics to develop closed loop controllers. Similar, [ThuruthelEtAl17] train neural networks and show that open loop predictive control can be performed with the learned dynamic model. While learning based methods can provide accurate modeling, a major downside is the data collection for the training procedure. Large datasets need to be generated in order to successfully learn the highly non-linear functions. Generating training data can be very expensive and costly, especially if not only a kinematic model but also dynamic model is learned. Furthermore, changes to the real-physical system can not easily be considered. Whereas for the aforementioned approaches parameters of the system can be updated and therefore changes can be taken into account easily, data-driven models need to undergo the complete training process again. Learning can take a long

time, especially for complex systems, and additionally the training data might have to be updated, when the real system is changed. In the same context, for a another even slightly different soft robot training data and the model can not be reused, potentially.

2.3 Inverse Direction

The reverse operation of finding a suitable input to reach a desire output is determined with the inverse model. While the forward direction can be completed by considering a kinematic model together with a dynamic model derivation, resulting in the equations of motion, the inverse direction is rather difficult to design due to infinite number of degrees of freedom. In terms of inverse kinematics, many works present different approaches to determine the system input for a soft robot. For example, [GiorelliEtAl12] model 2D kinematics using Cosserat rod theory and approximate the inverse kinematics by deriving a Jacobian. Results showed high computational cost and low accuracy, motivating the representation with feed-forward neural networks in [GiorelliEtAl13], extended in later work with an additional layer to consider initial manufacturing inaccuracies and long-term constant deflection of flexible materials [FangEtAl22]. However, completing the inverse model by considering inverse dynamics remains a current challenge in research. Therefore, representing the inverse dynamics remains an open topic in current research.

2.3.1 Inverse Dynamics for Softrobots

As described above, the inverse kinematics problem has been solved with different approaches, but only limited works exist, which consider representing dynamics in the inverse direction. [AmouriMahfoudiZaatri19] take the common parameterization of [WebsterJones10] and use Euler-Lagrange approach for deriving a dynamic model. However, their system is limited towards a single segment only with 2 degrees of freedom, and uses simplifications to derive the models. To solve the inverse dynamics problem, a simple Simulink system is implemented together with numerical derivatives of velocities and acceleration. A spatial example for a dynamic model is given by [JensenEtAl22], who design forward and inverse dynamics with the PCC assumption introduced in [AllenEtAl20] by using a recursive Newton-Euler approach, originating from rigid robotics field. However, their real robot is of a hybrid type, consisting of rigid links and pneumatically actuated soft joints. [ThuruthelRendaIida20] design a closed-loop controller for a soft robot by approximating the inverse dynamics controller with a neural network.

2.3.2 Inverse Dynamics for Underactuated Systems

The lack of existing approaches to represent the inverse dynamics gives reason to look at approaches from other research fields. Especially in the case of discretization into segments during the derivation of kinematic modeling, soft robots can be seen as flexible multibody systems. These usually belong to the family of underactuated multibody systems because their elastic deformation cannot be actuated directly [Drücker22]. This motivates to apply model inversion approaches from underactuated multibody systems theory to soft robots.

A general approach for deriving the input-output relationship of a non-linear underactuated system is given in [Seifried14], known as Byrnes-Isidori normal form. In this approach the system dynamics are transformed into the Byrnes-Isidori normal form. This new state space representation can be used to find an inverse model and determination of the system input. However, derivation of the Byrnes-Isidori normal form becomes laborious for multibody systems with a large number of degrees of freedom with multiple inputs and outputs. The output function has to be determined in algebraic form and the system needs to be present in input-affine form. Further, to bring the system into the Byrnes-Isidoris normal form a non-linear state transformation is required.

As an alternative to the Byrnes-Isidori normal form, the servo-constraints approach is a promising alternative to solving the inverse problem for the present system [Kirgetov67]. This method has been studied for significantly complex systems in [Drücker22]. On behalf of determining a control input strategy, the equations of motion describing the dynamic behavior are extended by constraints. These additional equations treat the system output as additional constraints on the system [BlajerKolodziejczyk04]. The systems governing equations together with the constraints appear as differential algebraic equations (DAE), which can be solved as a constrained problem using numerical algorithms [BlajerSeifriedKolodziejczyk15]. The advantage lies within combining the desire to force the system output to follow prescribed motion and determining the control input during the solving process at the same time. The servo-constraints realization is a simple and promising approach to handle the inverse direction of underactuated systems, such as soft robots.

Chapter 3

Forward Model

After the previous chapter introduced common approaches for deriving forward model for soft robots, this chapter contributes on establishing a representation for the forward model. First, one of the aforementioned methods for modeling kinematics is chosen in Sect. 3.1. The kinematic model is then derived in Sect. 3.2. Finally, Sect. 3.3 continues with determining the equations of motion to model the forward dynamics.

3.1 Modeling Choice for Kinematics

As shown in the previous chapter, there exist several popular methods on modeling forward kinematics of soft robots. Each method has their benefits and disadvantages. These approaches are summarized in the following and one of the methods is selected.

FEM provide very precise modeling, but come with the cost of expensive computation, and for real-time application complex workarounds have to be considered. The Cosserat rod theory models slender soft robots as flexible beams. They allow fast computation times and provide highly accurate results. PCC models are the most common choice because of their simplicity in application. However, singularity appearance and potential inaccuracy depending on the robots shape arise during application. Learning based strategies produce very accurate results together with low computation cost but rely on extensive data collection to train the models. Additionally, for dynamic modeling further networks are required in addition to the kinematic networks.

For this work the PCC-approach is chosen for deriving a model for forward kinematics, due to its simple yet effective representation. The present soft robot has a caterpillar-like shape, which shows promise on accurate modeling by segments with constant curvature.

3.2 Piecewise Constant Curvature Kinematics

The PCC-approach is one of the most common and oldest choice for modeling soft robot kinematics. The robot is discretized into N segments, as visualized in Fig. 3.1. The bending of each segment i is described by parameters locally. The transformation from one frame to another is given by a positional vector $\boldsymbol{p}_{i,\mathrm{loc}}$, pointing from the segment's origin towards the tip, and rotation matrix $\boldsymbol{R}_{i,\mathrm{loc}}$. Both quantities $\boldsymbol{p}_{i,\mathrm{loc}}$ and $\boldsymbol{R}_{i,\mathrm{loc}}$ are defined in the segment's local frame.

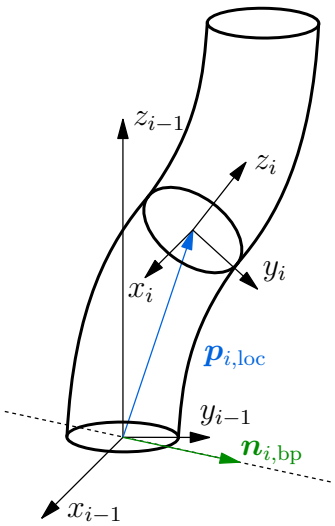


Figure 3.1: Modeling with the PCC-approach.

3.2.1 Parameterization

In this work two parameterization strategies for describing each of the $i = 1 \ldots N$ segments are examined. The first parameterization is introduced in [RoneBen-Tzvi14]. The authors parameterize each segment by separate curvatures β_i and γ_i in x and y direction, together with a third additional coordinate ϵ_i, which represents the twist angle. However, for this work the angle ϵ_i is neglected as experimental results showed it's negligible impact on the model [RoneBen-Tzvi14], [Wiek21]. The parameterization is visualized in Fig. 3.2.

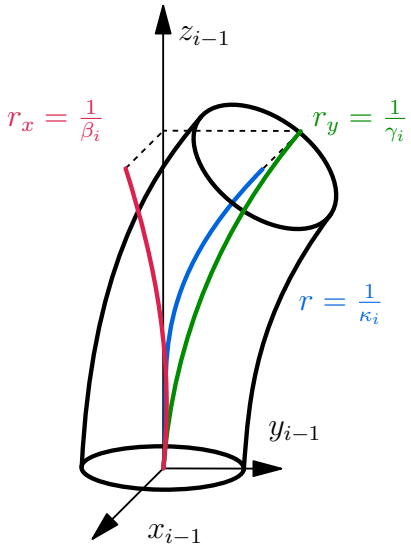


Figure 3.2: Parameterization according to [RoneBen-Tzvi14].

These three coordinates are then used to calculate intermediate coordinates, the total curvature κ_i, the bending angle φ_i and plane orientation θ_i, given by

$$\kappa_i = \sqrt{\beta_i^2 + \gamma_i^2}, \tag{3.1}$$
$$\varphi_i = \kappa_i L_i, \tag{3.2}$$
$$\theta_i = \arctan 2(\gamma_i, \beta_i). \tag{3.3}$$

The parameters are then used to derive the position vector $\boldsymbol{p}_{i,\text{loc}}$ and rotation matrix $\boldsymbol{R}_{i,\text{loc}}$. Together they describe the transformation between each segment.

While this parameterization is very clear and straightforward, it comes with a few disadvantages. First it contains singularities for certain configurations. The position vector $\boldsymbol{p}$ defines a singularity when a segment reaches the straight position, $\kappa_i = 0$. A division by zero appears in all three coordinates. The critical terms are

$$f_1(\varphi_i, \kappa_i) = \frac{(1 - \cos\varphi_i)}{\kappa_i}, \tag{3.4}$$
$$f_2(\varphi_i, \kappa_i) = \frac{\sin\varphi_i}{\kappa_i}. \tag{3.5}$$

The term f_1 appears in the x and y coordinates, f_2 in the z coordinate of $\boldsymbol{p}$. For zero curvature $\kappa_i = 0$ the bending angle becomes $\varphi_i = 0$ and in all entries of $\boldsymbol{p}_i$ a division of 0 by 0 occurs. Because this is a removable singularity the case of $\kappa_i = 0$ and $\varphi_i = 0$ can be considered separately. The robot is in its straight configuration then, where

$$\boldsymbol{p}_i = \begin{bmatrix} 0 & 0 & L_i \end{bmatrix}. \tag{3.6}$$

A second disadvantage is given by the $\arctan 2(y, x)$ operation used in Eq. (3.3). The function is not defined if the numerator and denominator both are zero. Many programming languages, such as MATLAB and PYTHON, assign a value to replace the singularity and provide a result once the operation is called with $x = 0$ and $y = 0$ (both return 0). But if additional trigonometric operations are called on top of the very operation, these languages tend to simplify the expression for computation speed, for example

$$\cos(\arctan 2(y, x)) = \frac{x}{\sqrt{x^2 + y^2}} \tag{3.7}$$

$$\sin(\arctan 2(y, x)) = \frac{y}{\sqrt{x^2 + y^2}}. \tag{3.8}$$

This leads to further singularities. To avoid the simplification, the $\cos(x)$ and $\sin(x)$ operation can be replaced by Taylor expansions at $x = 0$ again which results in a defined configuration at $\kappa = 0$ for the kinematics.

Third, for the derivation of the forward dynamics in Sect. 3.3, both, the position vector $\boldsymbol{p}$ and rotation matrix $\boldsymbol{R}$ have to be differentiated by the coordinate parameters, which again results in several terms possessing a division by κ. The reason for the appearing of a division by zero is the differentiation of $\arctan 2$, which results in

$$\frac{\partial}{\partial x} \arctan 2(y, x) = -\frac{y}{\sqrt{x^2 + y^2}}, \tag{3.9}$$

$$\frac{\partial}{\partial y} \arctan 2(y, x) = \frac{x}{\sqrt{x^2 + y^2}}. \tag{3.10}$$

These expressions are deeply nested throughout all of the derived terms necessary for the dynamics and replacing them by approximations for the case of a straight configuration becomes nearly impossible.

Because of this, [RoneBen-Tzvi14] and [Wiek21] add very small values to the parameters γ_i and β_i once they approach 0, in order to avoid the singularity. However, this results in a severe discontinuity in the function, which remains a problem for numerical solvers, as jumps in the function can be problematic to handle for solvers with variable step size control.

Because of the unsatisfying approaches to deal with the singularity occurrence, [AllenEtAl20] presented a non singular parameterization. It is derived from a single rotation around an axis from the base frame to the tip frame of a segment, which becomes advantageous as no plane rotation is required and therefore the differentiation of $\arctan 2$ is omitted. Their parameterization does include singularities for the straight configuration, too, but they can be replaced by Taylor expansion, again. This promises a reliable description of the kinematics and no nesting of singularity expression during the derivation of the dynamics in Sect. 3.3 evolves. Furthermore, [JensenEtAl22] chose the very representation for deriving dynamics for their robot.

[AllenEtAl20] introduce a rotation vector to paramterize each segment. It consists of positive rotations μ and ν around the x- and y-axis

$$\boldsymbol{w} = \begin{bmatrix} \mu_i & \nu_i & 0 \end{bmatrix}^T. \tag{3.11}$$

The length of a segment s is introduced as an additional parameter but not included in the rotation vector Eq. (3.11). However, two adjustments have been made for this work. First, the real robot model possesses a pattern in its body shape, which motivates to separate the robot into segments with fixed lengths $s = L_i$, chosen according to the dimensions. Second, because the rotation around the x-axis is defined as mathematically positive, negative values for μ will result in bending in positive x direction. To avoid this, μ is defined as a negative rotation around the x-axis in this work. The parameterization of [AllenEtAl20] is shown in Fig. 3.3.

The total bending angle of a segment can be obtained by

$$\varphi_i = \|\boldsymbol{w}\| = \sqrt{\mu_i^2 + \nu_i^2}. \tag{3.12}$$

The relationship between the length L_i and total rotation φ_i is given by

$$L_i = \varphi_i \|\boldsymbol{\rho}_{i,\mathrm{loc}}\|. \tag{3.13}$$

Here $\boldsymbol{\rho}$ describes the vector pointing at the center of the circle formed by the bent spine. It is given as

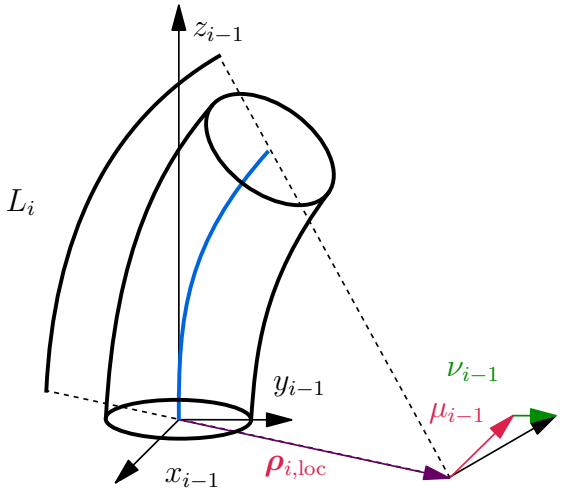


Figure 3.3: Parameterization according to [AllenEtAl20].

$$\boldsymbol{\rho}_{i,\mathrm{loc}} = \frac{L_i}{\varphi_i^2} \begin{bmatrix} \nu_i \\ \mu_i \\ 0 \end{bmatrix}. \tag{3.14}$$

With the appropriate parameters chosen the position vector is given by

$$\boldsymbol{p}_{i,\mathrm{loc}} = \begin{bmatrix} -\rho_{i,\mathrm{loc},x}\sigma_i \\ -\rho_{i,\mathrm{loc},y}\sigma_i \\ \|\boldsymbol{\rho}_{i,\mathrm{loc}}\| S_{\varphi_i} \end{bmatrix} \tag{3.15}$$

with $\rho_{i,\mathrm{loc},x}$ and $\rho_{i,\mathrm{loc},y}$ being the x and y entries of $\boldsymbol{\rho}_{i,\mathrm{loc}}$. Then the single rotation in the bending plane is accomplished by multiplication with the rotation matrix

$$\boldsymbol{R}_{i,\mathrm{loc}} = \begin{bmatrix} \sigma_i\bar{\nu}_i^2 + 1 & \sigma_i\bar{\mu}_i\bar{\nu}_i & \bar{\nu}_i \sin\varphi_i \\ \sigma_i\bar{\mu}_i\bar{\nu}_i & \sigma_i\bar{\mu}_i^2 + 1 & \bar{\mu}_i \sin\varphi_i \\ -\bar{\nu}_i \sin\varphi_i & -\bar{\mu}_i \sin\varphi_i & \cos\varphi_i \end{bmatrix} \tag{3.16}$$

with $\sigma_i = \cos(\varphi_i) - 1$, $\bar{\nu}_i = \dfrac{\nu_i}{\varphi_i}$ and $\bar{\mu}_i = \dfrac{\mu_i}{\varphi_i}$. Thus, the vector of generalized coordinates of a single segment is given by

$$\boldsymbol{y}_{i,\mathrm{loc}} = \begin{bmatrix} \mu_i & \nu_i \end{bmatrix}^T. \tag{3.17}$$

Here, $\boldsymbol{y}$ represents the general generalized coordinate vector

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_{1,\mathrm{loc}}^T & \boldsymbol{y}_{2,\mathrm{loc}}^T & \cdots & \boldsymbol{y}_{N,\mathrm{loc}}^T \end{bmatrix}^T. \tag{3.18}$$

The previously mentioned singularities appear for a straight segment configuration $\mu_i = 0$, $\nu_i = 0$ and therefore $\varphi_i = 0$, again, where both the position vector $\boldsymbol{p}_{i,\mathrm{loc}}$ and rotation matrix $\boldsymbol{R}_{i,\mathrm{loc}}$ possess zero divisions in almost every entry. However, all of these zero divisions appear in the form of

$$g_1(\varphi) = \frac{(\cos\varphi - 1)}{\varphi^2} \tag{3.19}$$

$$g_2(\varphi) = \frac{\sin\varphi}{\varphi} \tag{3.20}$$

which can be approximated by a Taylor expansion at $\varphi = 0$. It should be noticed, that even though the $\cos\varphi$ of $\boldsymbol{R}_{i,\mathrm{loc}}^{(3,3)}$ does not bother in the kinematic relationship, the cos function does create a zero division during the differentiation. But again, this term can be approximated with a Taylor series, too. This completes the full representation of the kinematic relationship as well as solidifies a basis for the derivation of the dynamic relationships in Sect. 3.3

3.2.2 Global Frame

Now that a complete parameterization for a single segment is established, a transformation of all quantities into a global coordinate frame is beneficial. The global frame is placed on the center axis at the point where the soft robot sticks out of its socket. The origin of the first segment coincidences with the global frame.

To express the rotation and position of each segment in the global frame, a recursive calculation can be performed. The local rotation matrix $\boldsymbol{R}_{i,\mathrm{loc}}$ and local position vector $\boldsymbol{p}_{i,\mathrm{loc}}$ expressed in each segments frame i are calculated for $i = 0 \ldots N$. Then the global rotation and position of a single segment can be obtained by

$$\boldsymbol{R}_i = \begin{cases} \boldsymbol{R}_{i,\mathrm{loc}} & \text{for} \quad i = 1 \\ \boldsymbol{R}_{i-1} \cdot \boldsymbol{R}_{i,\mathrm{loc}} & \text{for} \quad i > 1 \end{cases} \tag{3.21}$$

and

$$\boldsymbol{p}_i = \begin{cases} \boldsymbol{p}_{i,\mathrm{loc}} & \text{for} \quad i = 1 \\ \boldsymbol{p}_{i-1} + \boldsymbol{R}_{i-1} \cdot \boldsymbol{p}_{i,\mathrm{loc}} & \text{for} \quad i > 1 \end{cases} \tag{3.22}$$

for $i = 0 \dots N$. Each segments origin frame coincidences with its predecessors tip frame. The global frame is denoted with $\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{z}_0$. For the actual implementation of the recursive algorithm the position vector $\boldsymbol{p}_0$ and rotation matrix $\boldsymbol{R}_0$ of the global frame are chosen as

$$\boldsymbol{p}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{3.23}$$

$$\boldsymbol{R}_0 = \boldsymbol{I} \tag{3.24}$$

with $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ being the identity matrix.

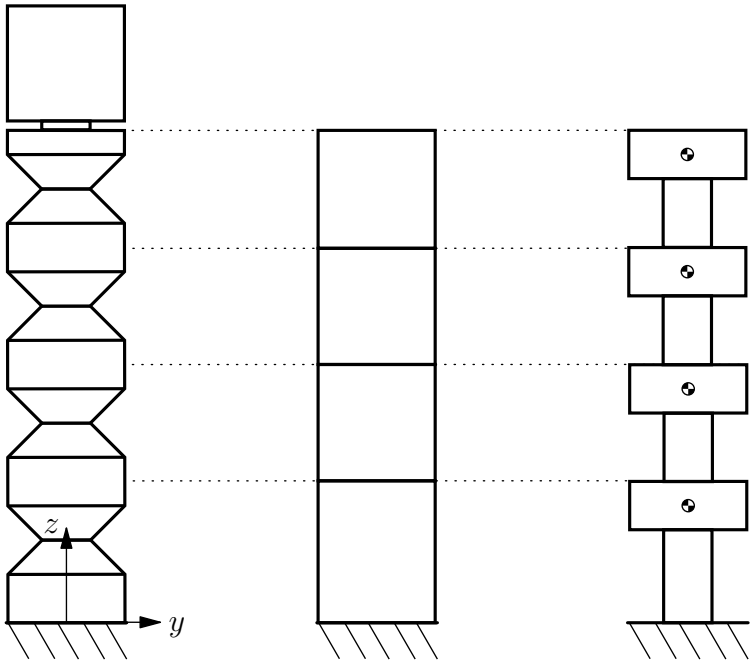
3.3 Forward Dynamics

In order to define a mapping between actuators and the generalized coordinate vector, the dynamic behavior of the soft robot has to be described. This can be accomplished by deriving the equations of motion, which represent the dynamic behavior of a mechanical system. Following the Newton-Euler procedure [SchiehlenEberhard20], the differential equations are determined by considering the dynamic reaction of the system to acting loads.

For this purpose, each segment can be seen as part of a multi-body system and it's dynamic behavior represented by a differential equation. Fig. 3.4 displays a simple sketch of the real robot together with the model used for parameterization, and visualization of a dynamic model. The real robot possesses caterpillar-like shape. The largest deformation will occur at the notches between the barrels. Therefore, the number of segments N can be chosen, so that each segment contains the notches in the middle, as visualized in Fig. 3.4. The dynamic model then adds a virtual disk, that is characterized by a mass and inertia.

3.3.1 External Loads

External loads describe forces and moments, which effect the mechanical system from outer sources. For the present soft robot the external loads consist of the gravitational force only. The gravitational force of a segment always points in negative $\boldsymbol{z}$ direction in the global coordinate frame, as the soft robot is placed in a socket that is fixed to the ground. With the mass m_i of the segment i and the gravitational acceleration g the gravitational force is described by



(a) (b) (c)

Figure 3.4: Visualization of the system: the real robot simplified (a), sketch of the parameterization model (b) and dynamic model (c).

$$\boldsymbol{f}_{g,i} = \begin{bmatrix} 0 & 0 & -m_i g \end{bmatrix}^T . \tag{3.25}$$

3.3.2 Actuation Loads

The actuation of the soft robot is achieved with cables attached to the soft robot's tip. Because the cables are guided through bolt holes in the robot's segments each segment underlies forces and torques, induced when the robot is bend by pulling the cables. Each segment has six guides in total, where cables can be pulled through. For this work only three cables are deployed, which results in six forces acting on a single segment, three at the upper holes u and three others at the lower holes d. The position of the cable guide holes on the downwards surface of a single segment in the local coordinate frame can be described by

$$\boldsymbol{r}^d_{i,1,\mathrm{loc}} = \begin{bmatrix} r_{\mathrm{holes}} & 0 & -l_{\mathrm{holes}}/2 \end{bmatrix}^T \tag{3.26}$$

$$\boldsymbol{r}^d_{i,2,\mathrm{loc}} = \begin{bmatrix} -r_{\mathrm{holes}}/2 & r_{\mathrm{holes}}\sqrt{3}/2 & -l_{\mathrm{holes}}/2 \end{bmatrix}^T \tag{3.27}$$

$$\boldsymbol{r}^d_{i,3,\mathrm{loc}} = \begin{bmatrix} -r_{\mathrm{holes}}/2 & -r_{\mathrm{holes}}\sqrt{3}/2 & -l_{\mathrm{holes}}/2 \end{bmatrix}^T . \tag{3.28}$$

The cable guide holes on the upwards surface are given by

$$\boldsymbol{r}^u_{i,1,\mathrm{loc}} = \begin{bmatrix} r_{\mathrm{holes}} & 0 & l_{\mathrm{holes}}/2 \end{bmatrix}^T \tag{3.29}$$

$$\boldsymbol{r}^u_{i,2,\mathrm{loc}} = \begin{bmatrix} -r_{\mathrm{holes}}/2 & r_{\mathrm{holes}}\sqrt{3}/2 & l_{\mathrm{holes}}/2 \end{bmatrix}^T \tag{3.30}$$

$$\boldsymbol{r}^u_{i,3,\mathrm{loc}} = \begin{bmatrix} -r_{\mathrm{holes}}/2 & -r_{\mathrm{holes}}\sqrt{3}/2 & l_{\mathrm{holes}}/2 \end{bmatrix}^T . \tag{3.31}$$

Here,the quantity r_{holes} represents the radius of the circle around the central axis, on which the holes are positioned, and l_{holes} denotes the length of the drillt holes. The position of the bolt holes is displayed in Fig. 3.5.

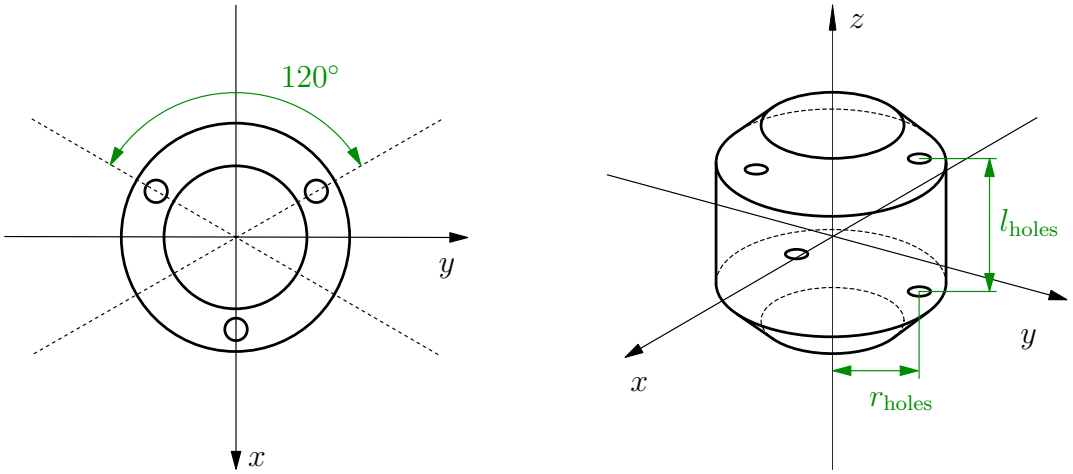


Figure 3.5: Positions of the bolt holes.

Then the positions of the cable guide holes $\boldsymbol{r}^u_{i,q}$ and $\boldsymbol{r}^d_{i,q}$ with $q = 1 \ldots 3$ can be transformed to the global frame by

$$\boldsymbol{r}^d_{i,q} = \boldsymbol{p}_i + \boldsymbol{R}_i \cdot \boldsymbol{r}^d_{i,q,\mathrm{loc}} \tag{3.32}$$

$$\boldsymbol{r}^u_{i,q} = \boldsymbol{p}_i + \boldsymbol{R}_i \cdot \boldsymbol{r}^u_{i,q,\mathrm{loc}} \tag{3.33}$$

with $q \in \{1, 2, 3\}$ denoting the according hole. In order to calculate the forces induced by pulling the cables, the effective direction has to be determined first. Each segment experiences two forces for each cable, a force acting towards the previous segment and a force acting towards the following segments. To obtain the effective direction, the positional vectors of two segments pointing towards the holes can be utilized, which guide the same cable.

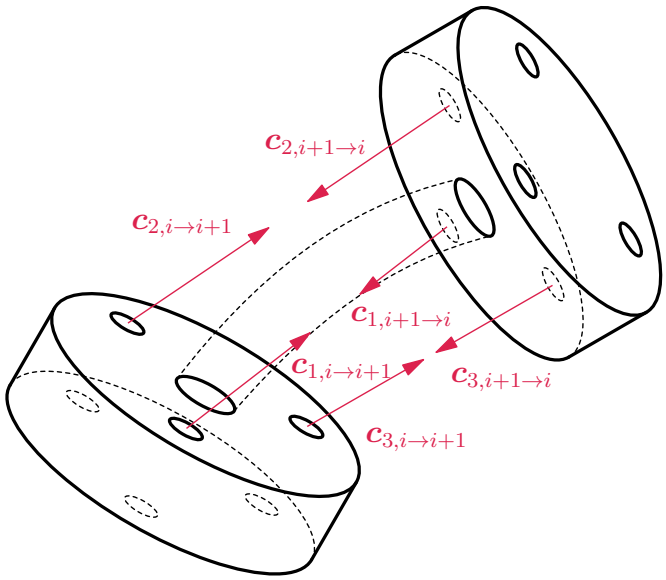


Figure 3.6: Separation into the different operation spaces

Then the normalized direction vector for the cable force towards the previous segment becomes

$$\boldsymbol{c}_{i\to i-1,q} = \frac{\boldsymbol{r}^u_{i-1,q} - \boldsymbol{r}^d_{i,q}}{\|\boldsymbol{r}^u_{i-1,q} - \boldsymbol{r}^d_{i,q}\|} \tag{3.34}$$

and towards the following segment

$$\boldsymbol{c}_{i\to i+1,q} = \frac{\boldsymbol{r}^d_{i+1,q} - \boldsymbol{r}^u_{i,q}}{\|\boldsymbol{r}^d_{i+1,q} - \boldsymbol{r}^u_{i,q}\|}. \tag{3.35}$$

For the last segment there is only forces acting towards the previous segment and therefore Eq. (3.35) can be set to

$$\boldsymbol{c}_{N\to N+1,q} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T. \tag{3.36}$$

With the normalized direction vectors the forces acting towards base direction can be calculated by

$$\boldsymbol{f}_{i\to i-1,\text{cab}} = \sum_{q=1}^{3}\left(\boldsymbol{c}_{i\to i-1,q}\cdot\sum_{n=i}^{N}F_{n,q}\right). \tag{3.37}$$

Here $F_{n,q}$ describes the cable force value acting on the segment i at the hole q. Note that the index for the second sum starts at the very segment i. All forces acting on the segment i and the following ones are included in the sum.
Similarly, the force pulling the segment i towards the tip direction is induced by all cable forces of the following segments $i+1\ldots N$ and is calculated by

$$\boldsymbol{f}_{i\to i+1,\text{cab}} = \sum_{q=1}^{3}\left(\boldsymbol{c}_{i\to i+1,q}\cdot\sum_{n=i+1}^{i+2}F_{n,q}\right). \tag{3.38}$$

Because the cable forces are not located in the center of every segment each force induces a torque around the center of mass. The torque caused by the cable forces originating for the previous segment $i-1$ can be obtained by

$$\boldsymbol{\ell}_{i\to i-1,\text{cab}} = \sum_{q=1}^{3}\left(\left(\boldsymbol{R}_i\cdot\boldsymbol{r}_{i,q}^{d}\right)\times\left(\boldsymbol{c}_{q,i\to i-1}\cdot\sum_{n=i}^{N}F_{n,q}\right)\right) \tag{3.39}$$

and the torque caused by the cable forces originating from the following segment is calculated with

$$\boldsymbol{\ell}_{i\to i+1,\text{cab}} = \sum_{q=1}^{3}\left(\left(\boldsymbol{R}_i\cdot\boldsymbol{r}_{i,q}^{u}\right)\times\left(\boldsymbol{c}_{q,i\to i+1}\cdot\sum_{n=i+1}^{N}F_{n,q}\right)\right). \tag{3.40}$$

3.3.3 Internal Loads

During actuation the cables force the robot to bend, which causes the elastic material to deform. This deformation causes internal torques, which react to the cable forces. Both bending torques and torsional torques appear in theory, but because of the limited actuation with three cables, allowing bending in $x-y$ direction only, very small torsional effects occur. In the work of [RoneBen-Tzvi14] and [Wiek21] the torsional loads have been taken into account, given in the form of

$$\ell_{i,\text{tor}} = GI_{\text{loc},zz}\cdot\frac{\epsilon_i}{L_i}. \tag{3.41}$$

The shear modulus of the elastic material, which the robot consists of, is denoted by G and $I_{\text{loc},zz}$ is the inertia around the z axis. But the evaluation showed

little influence of torsion during bending, which is why it is neglected in this work.

According to [GrossEtAl09] a homogeneous beam, which bends with total curvature κ_i, experiences a torque depending on κ_i. Therefore the pure torque value for a single segment is given by

$$\ell_{i,\text{bend}} = EI_{\text{loc},xx} \cdot \kappa_i = EI_{\text{loc},xx} \cdot \frac{\sqrt{\mu_i^2 + \nu_i^2}}{L_i} \tag{3.42}$$

with E denoting the Young's module of the elastic material and $I_{\text{loc},xx}$ being the inertia around the x-axis. Considering the free body diagram of a single elastic segment both ends experience bending by the acting torques but with a different sign. Therefore, $\ell_{\text{bend},i}$ acts on the current segment as well as on the previous segment. Then the vector representations of the bending torques, which act on two disks connected to the same elastic link, can be obtained by

$$\boldsymbol{\ell}_{i,\text{bend}} = -\ell_{i,\text{bend}} \cdot \boldsymbol{n}_{i,\text{bp}} \tag{3.43}$$

$$\boldsymbol{\ell}_{i+1\to i,\text{bend}} = \ell_{i+1,\text{bend}} \cdot \boldsymbol{n}_{i+1,\text{bp}}. \tag{3.44}$$

The equations include the normal of the bending plane, visualized in Fig. 3.1 of a single segment represented in the global frame

$$\boldsymbol{n}_{i,\text{bp}} = \boldsymbol{R}_{i-1} \cdot \begin{bmatrix} -\sin(\theta_i) & \cos(\theta_i) & 0 \end{bmatrix}^T. \tag{3.45}$$

Eq. (3.45) includes the rotation angle to the bending plane $\theta_i = \arctan 2(\mu_i, \nu_i)$, which turned out to be challenging in combination with another trigonometric function, as seen in Eq. (3.7) and Eq. (3.8), respectively. But because this hurdle appears in this equation only and no further relationship is derived from Eq. (3.45) the cos and sin functions can be replaced by their Taylor approximations.

Lastly, damping can be considered for the given system. Commonly, soft robots show rather high damping [ThuruthelRendaIida20]. To investigate on the damping, the present soft robot is moved out of its state of equilibrium by hand. Once the robot is bend far enough, the robot is let go and the position of the tip is tracked. The measured tip positions are depicted in Fig. 3.7.

The damping ratio δ can be determined by considering the measured amplitudes of the oscillation. Here, the first four amplitudes of two successive peaks are

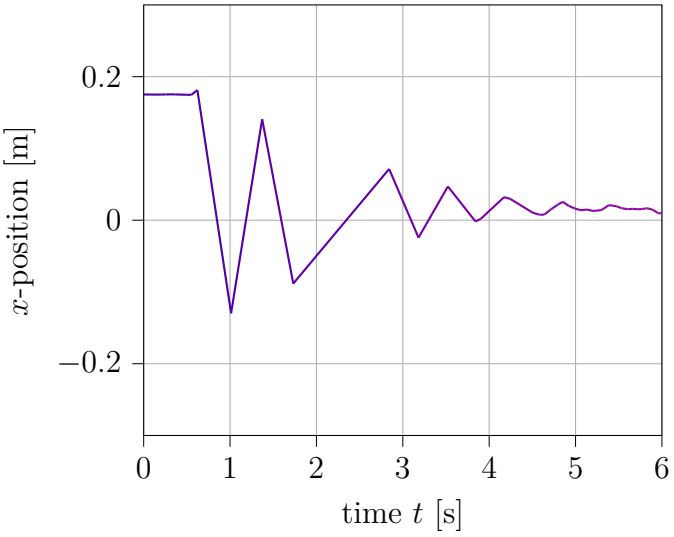


Figure 3.7: Measured position of the robot's for free vibration.

considered. In order to determine the damping ratio δ the logarithmic decrement ζ is considered. It is computed by

$$\zeta = \ln\left(\frac{y_1 - y_3}{y_2 - y_4}\right). \tag{3.46}$$

The amplitudes y_{1-4} can be read from the graph and the value for ζ computed to

$$\zeta = \ln\left(\frac{0.179\,\mathrm{m} - 0.1391\,\mathrm{m}}{0.1297\,\mathrm{m} - 0.0198\,\mathrm{m}}\right) = 0.0199. \tag{3.47}$$

Then the damping ratio is computed by

$$\delta = \frac{\zeta}{\sqrt{\zeta^2 + (2\pi)^2}} \tag{3.48}$$

The result shows a rather small damping ratio of $\delta = 0.00317$ for the present soft robot, which is why damping is neglected in this work.

3.3.4 Equations of Motion

At last, the inertia tensor $\boldsymbol{I}_i$ of each segment i is required for the derivation of the equations of motion. It is given by

$$\boldsymbol{I}_i = \boldsymbol{R}_i^T \cdot \begin{bmatrix} I_{\text{loc},xx} & 0 & 0 \\ 0 & I_{\text{loc},xx} & 0 \\ 0 & 0 & I_{\text{loc},zz} \end{bmatrix} \cdot \boldsymbol{R}_i. \tag{3.49}$$

The equations of motion are obtained by following the Newton-Euler formalism. The Newton-Euler equations are derived locally for each segment in the global frame and then reassembled into the form

$$\boldsymbol{M}(\boldsymbol{y}, t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) = \boldsymbol{q}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{u}. \tag{3.50}$$

Here, $\boldsymbol{M}(\boldsymbol{y}, t) \in \mathbb{R}^{f \times f}$ denotes the symmetric mass matrix, $\boldsymbol{k}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) \in \mathbb{R}^f$ represents the the Coriolis, centrifugal and gyroscopic forces and $\boldsymbol{q}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) \in \mathbb{R}^f$ are the applied forces, with $f = 2 \cdot N$. This representation includes the input $\boldsymbol{u} \in \mathbb{R}^m$ to the system together with the input distribution matrix $\boldsymbol{B}(\boldsymbol{y}) \in \mathbb{R}^{f \times m}$ providing a mapping between the input values and generalized coordinates. In other words, $\boldsymbol{B}(\boldsymbol{y})$ is the sum of all cable force and torque vectors. Then, the input $\boldsymbol{u}$ consists of $m = 3$ forces

$$\boldsymbol{u} = \begin{bmatrix} F_{\text{cab},1} & F_{\text{cab},2} & F_{\text{cab},3} \end{bmatrix}^T, \tag{3.51}$$

which are induced by the cables used for actuation. It is assumed that the cable forces stay consistent and each hole experiences the same absolute force value being the total cable force, without additional consideration of friction loss.

3.4 Actuation

In this work only three cables are used for actuation. Therefore, the soft robot can only bend one directional, which allows only C-shapes. To achieve bending, only two of the three cables need to be pulled. The third one has to be lose to allow the movement. For this purpose, an additional input-mapping is required. For the present work, the robot's workspace is divided into three pieces. Each piece lies between two bolt holes. The according cables, which are routing through these holes, are the actuation inputs for the very workspace. The workspace distribution is characterized with the addition of an angle ψ defined in the x-y plane. A mapping can then be established as

$$\boldsymbol{u} = \begin{bmatrix} F_{\text{cab},1} & F_{\text{cab},2} & F_{\text{cab},3} \end{bmatrix}^T = \boldsymbol{\Phi}^T \cdot \boldsymbol{u}_{\text{red}} \tag{3.52}$$

with $\boldsymbol{u}_{\text{red}}$ denoting the reduced unknown input consisting of the two active forces. The mapping matrix $\boldsymbol{\Phi}$ is dependent on the location of the desired tip position, given by

$$\boldsymbol{\Phi} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \text{for} \quad \psi \in [0, 120°) \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \text{for} \quad \psi \in [120°, 240°) \\ \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} & \text{for} \quad \psi \in [240°, 360°) \end{cases} \tag{3.53}$$

with $\psi = \arctan 2(\tilde{z}_y, \tilde{z}_x)$. The two values $\tilde{z}_y$ and $\tilde{z}_x$ are the x and y coordinates of the desired output position of the end-effector, explained in Sect. 3.5. For example, if $\psi = 90°$ the force input to the system is $\begin{bmatrix} \boldsymbol{u}_{\text{red}} & 0 \end{bmatrix}^T$. The actuation areas for each motor are visualized in Fig. 3.8.

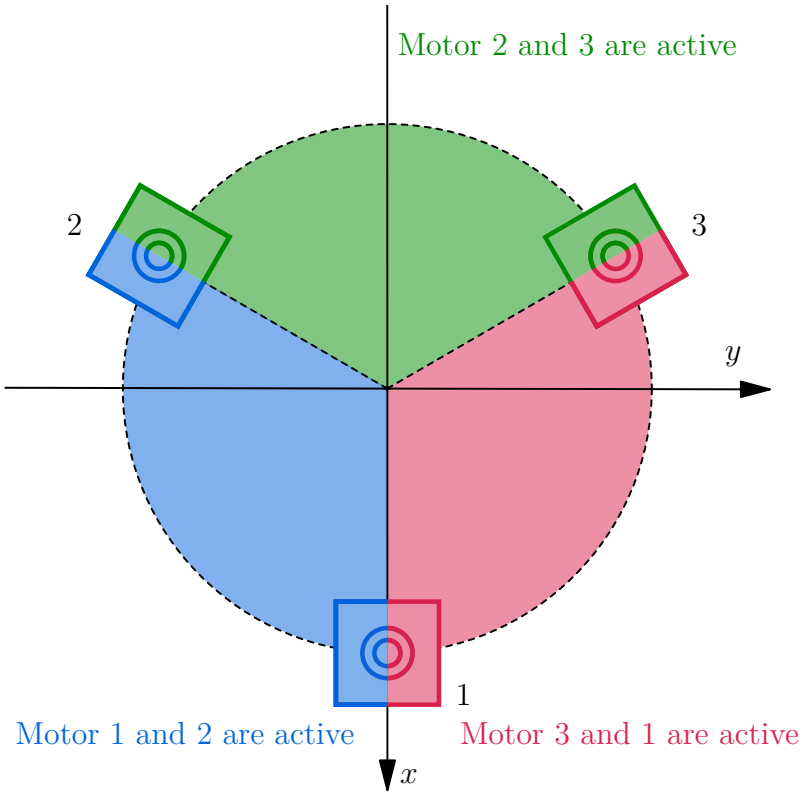


Figure 3.8: The three areas and active motors.

The three cables, which are attached to the robot, are pulled by three motors. If the current state $\boldsymbol{y}$ is known the required change in length for each cable can be

computed by the kinematics. It is determined by evaluating the robot's shape at the current state $\boldsymbol{y}$ and the positions of the bolt holes in space. The denominator in Eq. (3.35) gives the current distance between the bolt holes for the robot with its current shape at $\boldsymbol{y}$. Then the change in length of a single cable $q = 1 \dots 3$ is obtained with

$$\Delta l_{K,q} = l_{\text{cab},\boldsymbol{y}_0} - \sum_{n=0}^{N} (\|\boldsymbol{r}^d_{n+1,q} - \boldsymbol{r}^u_{n,q}\|)|_{\boldsymbol{y}}. \tag{3.54}$$

However in practice, the cables and the robot itself can stretch under load. Therefore, simple kinematic actuation is not sufficient enough and the elasticity has to be taken into account. This is achieved by modeling a virtual spring. The force of a spring can be obtained by considering Hooke's law

$$F_{\text{cab},q} = \Delta l_{F,q} \cdot c_q \tag{3.55}$$

with $\Delta l_{F,q}$ denoting the change in length of the virtual spring, here the change in cable length, and c_q a positive real value, the spring coefficient. Once c_q is determined experimentally, the required change in cable length can be calculated in order to exert a certain force on the robot. The total change in cable length is then the addition

$$\Delta l_q = \Delta l_{F,q} + \Delta l_{K,q}. \tag{3.56}$$

For the latter actuation of the real physical robot in Chap. 6, Δl_q has to be converted to the necessary angular rotation α_q of the motors. This is accomplished by

$$\alpha_q = \frac{2\Delta l_q}{d_{\text{reel}}}, \tag{3.57}$$

where d_{reel} denotes the diameter of the reels mounted to the motors.

3.5 Output

Finally, a system output $\boldsymbol{z}$ can be chosen, which completes the forward direction of the system. The output is defined by a function $\boldsymbol{h}$, which sets the relationship between the generalized coordinates $\boldsymbol{y}$ and the desired output $\boldsymbol{z}$

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{y}). \tag{3.58}$$

To achieve accurate trajectory tracking, an appropriate output has to be chosen that can represent the workspace of the softrobot. For example, rigid robotic arms with defined joints are commonly used to approach chosen points in the working area of the robot and therefore an output representation of positional information of the end effector is favorable

$$\boldsymbol{z} = \begin{bmatrix} x_N & y_N & z_N \end{bmatrix}^T. \tag{3.59}$$

However, for the present soft robot the operation space is limited to a cropped spatial ellipse around the robots base Fig. 3.9. This is due to the fact, that in its current configuration the robot will take the shape of a half circle only, when actuated. The limitation to three cables allows no S-shaped configuration, which would be required to reach any given point in a spacial working area. With only three cables attached only two degrees of freedom are available resulting in an operation space in form of a hull around the robots base.

In order to achieve accurate trajectory following, the output $\boldsymbol{z}$ should return an appropriate output representation. A straightforward representation would be to consider the total rotations μ and ν as the output values directly with $\boldsymbol{h}$ being a unity matrix. However, this would require a definition of the desired trajectory with the same representation, which will be laborious to determine. Another promising approach is the representation of the tip position with spherical coordinates because of the almost spherical shaped task space, defined by

$$r_p = \sqrt{x^2 + y^2 + z^2} \tag{3.60}$$

$$\varphi_p = \arccos \frac{z}{r_p} \tag{3.61}$$

$$\theta_p = \arctan 2(\mu, \nu). \tag{3.62}$$

The spherical coordinates come with the advantage of reusing the already previously applied definition for the plane orientation θ_p. However, since the workspace is limited to an elliptical hull the spherical coordinates are rather complicated, as only two degrees of freedom limit the choice for the coordinates. Choosing one of the coordinates as constant, e.g. the radius r_p will result in a round hull rather than a spherical one. Therefore spherical coordinates do not provide any significant advantage.

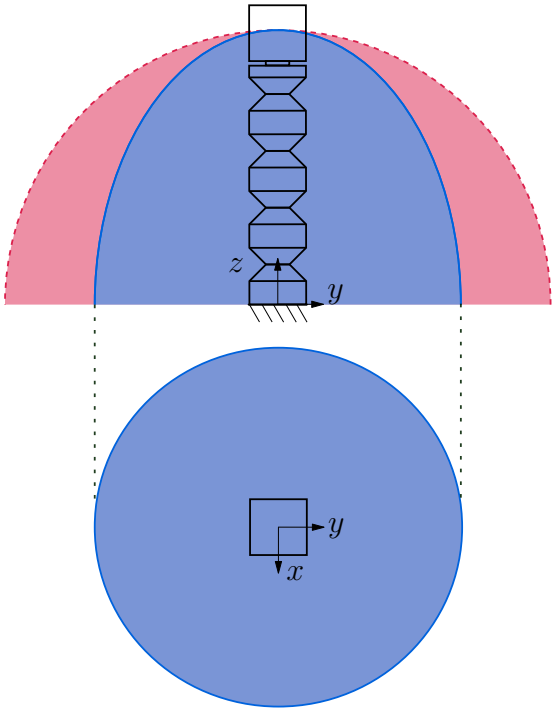


Figure 3.9: Elliptical workspace and workspace described by spherical coordinates for constant radius r_p.

Instead, another output representation can be considered by fixing the output to an x-y projection of the hull, seen in the lower part of Fig. 3.9. When looking from a top-down view each point on the hull can be described by the x and y parts of the global position vector of the soft robot's end-effector. The position in z is then given by choosing x and y. This results in an output given by

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{y}) = \begin{bmatrix} p_{x,N}(\boldsymbol{y}) & p_{y,N}(\boldsymbol{y}) \end{bmatrix}^T . \tag{3.63}$$

It should be noted, that the position vector of the last segments end might be not sufficient depending on the design of the actual system. In this work the robot is equipped with a cube used as an end-effector. The cube will be tracked as explained in Chap. 5. Because of that $\boldsymbol{z}$ will be extended by considering the additional length in z_N in the local frame of the last segment.

Chapter 4

Inverse Model

This chapter addresses the derivation of the inverse model, which is required to determine the necessary system input for reaching a desired output. For this purpose, the servo-constraints approach is chosen, which is introduced in Sect. 4.1. Afterwards, the system is completed by incorporating the servo-constraints framework into the equations of motion in Sect. 4.2.

4.1 Inverse Model with Servo-Constraints

With the equations of motion Eq. (3.50) completing the forward mapping between joint space and configuration space, an inverse relationship is derived to complete the model. Given an output $\boldsymbol{z}$ the according input $\boldsymbol{u}$ is determined, in order to achieve feedforward control for trajectory tracking. The servo-constraints approach allows for a representation of the inverse direction.

4.1.1 Basic Concept

In general, servo-constraints are treated as motion specification on a system and have to be distinguished from classical contact and passive constraints, such as hard surfaces or rigid joints and links [BlajerSeifriedKolodziejczyk15]. The constraints $\boldsymbol{s}(\boldsymbol{y}, t)$ are defined on positional, velocity and acceleration level, by enforcing the output of the system to be equal to the desired output motion, which is given by

$$\boldsymbol{s}(\boldsymbol{y}, t) = \boldsymbol{h}(\boldsymbol{y}) - \tilde{\boldsymbol{z}}(t) = \boldsymbol{0} \tag{4.1}$$

$$\dot{\boldsymbol{s}}(\boldsymbol{v}, \boldsymbol{y}, t) = \boldsymbol{H}(\boldsymbol{y})\boldsymbol{v} - \dot{\tilde{\boldsymbol{z}}}(t) = \boldsymbol{0} \tag{4.2}$$

$$\ddot{\boldsymbol{s}}(\dot{\boldsymbol{v}}, \boldsymbol{v}, \boldsymbol{y}, t) = \boldsymbol{H}\dot{\boldsymbol{v}} + \dot{\boldsymbol{H}}\boldsymbol{v} - \ddot{\tilde{\boldsymbol{z}}}(t) = \boldsymbol{0}. \tag{4.3}$$

Here, the specified output position, velocity and acceleration are denoted as $\tilde{\boldsymbol{z}}(t)$, $\dot{\tilde{\boldsymbol{z}}}(t)$ and $\ddot{\tilde{\boldsymbol{z}}}(t)$, respectively. For the servo-constraints on velocity and acceleration level, the output function $\boldsymbol{h}$ needs to be differentiated twice, resulting in the Jacobi matrix $\boldsymbol{H} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{y}}$ and its derivative by time $\dot{\boldsymbol{H}}$. Here, $\boldsymbol{v}$ denotes the differentiation of $\boldsymbol{y}$ with respect to time and $\dot{\boldsymbol{v}}$ the further derivative of $\boldsymbol{v}$ with respect to time.

The soft robot considered in this work is an underactuated system. This becomes clear when the dimension of the input is considered,

$$m = \dim(\boldsymbol{u}_{\mathrm{red}}) = 2 \leq f. \tag{4.4}$$

This is the case, if the robot is described with more than one segment in the PCC. Only in the case of $N = 1$ the system is fully actuated. Consequently, the motion of the system is fully specified for a parameterization with only one segment ($f = 2$) and partly specified for any other case

$$\dim(\boldsymbol{z}) = 2 \leq f. \tag{4.5}$$

Under the assumption of a fully actuated system and therefore fully specified motion, an input-output relationship can be derived by reassembling the servo-constraints and addition of the equations of motion Eq. (3.50) to

$$\tilde{\boldsymbol{u}}(t) = (\tilde{\boldsymbol{H}}\tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{B}})^{-1}(\ddot{\tilde{\boldsymbol{z}}}(t) - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{M}}^{-1}(\tilde{\boldsymbol{q}} - \tilde{\boldsymbol{k}}) - \tilde{\boldsymbol{h}}). \tag{4.6}$$

The required input $\tilde{\boldsymbol{u}}(t)$ is directly related to the specified output acceleration $\ddot{\tilde{\boldsymbol{z}}}(t)$, where the tilde denotes quantities evaluated at $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{v}}$ and t. The $f \times f$ matrix $\boldsymbol{Y}(\boldsymbol{y}) = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{B}}$ is assumed to be invertible for the set assumptions and a control law can be established straightforward [BlajerSeifriedKolodziejczyk15].

As mentioned above, the soft robot is an underactuated system for the case of choosing more than a single segment to describe the robots configuration. Then the matrix $\boldsymbol{Y}(\boldsymbol{y})$ is of size $m \times m$ and can be of maximal rank or can

be rank deficient. In the case of maximum rank all m specified outputs can be actuated by the m system inputs directly. Internal dynamics remain in the system, which are influenced by the inverse dynamics control. These can be of stable nature (minimum phase system) or destabilize the underactuated system in partly specified motion (non-minimum phase system) [Tomlin01]. For rank deficiency, the outputs are not directly actuated by the system inputs and the servo-constraints have to be designed through dynamical couplings in the system instead [BlajerSeifriedKolodziejczyk15].

To realize servo-constraints for the given system, first the stability of the internal dynamics is examined. From a control theory perspective an eigenvalue problem is analyzed, which provides important information on the stability of the internal dynamics. Second, the input-output relationship will show if rank deficiency is present for the matrix $\boldsymbol{Y}$ and how the servo-constraints can be designed.

4.1.2 Stability Analysis of Internal Dynamics

In classic control theory, single-input single-ouput (SISO) systems are characterized by the transfer function G resembling the relationship between system input and output. In the frequency domain $s \in \mathbb{C}$ this relationship is defined by

$$G(s) = \frac{Y(s)}{U(s)} = \frac{p_n(s)}{p_d(s)} \tag{4.7}$$

where $U(s)$ marks the system input and $Y(s)$ the system output. The transfer function is a fraction of the numerator and denominator polynomials $p_n(s)$ and $p_d(s)$, whose roots are called zeros and poles of $G(s)$, respectively. Consequently, the inverse model is given by

$$U(s) = G(s)^{-1}Y(s) = \frac{p_d(s)}{p_n(s)}Y(s), \tag{4.8}$$

where the poles of the forward dynamics become the zeros of the inverse dynamics. Therefore, if the zeros of the forward dynamics are in the left-half plane of the root locus diagram the inverse dynamics are minimum phase and the inverse systems possesses stable internal dynamics.

For the local analysis of multiple-input multiple-output (MIMO) system, as the present soft robot, the analysis begins with the linearization of the equations of motion at a stationary linearization point [Seifried14]

$$\begin{bmatrix} \dot{\boldsymbol{y}}_{\mathrm{eq}} \\ \dot{\boldsymbol{v}}_{\mathrm{eq}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{\mathrm{eq}} \\ \boldsymbol{M}(\boldsymbol{y}_{\mathrm{eq}})^{-1} \left(\boldsymbol{q}(\boldsymbol{y}_{\mathrm{eq}}, \boldsymbol{v}_{\mathrm{eq}}) - \boldsymbol{k}(\boldsymbol{y}_{\mathrm{eq}}, \boldsymbol{v}_{\mathrm{eq}}) + \boldsymbol{B}(\boldsymbol{y}_{\mathrm{eq}}) \boldsymbol{u}_{\mathrm{eq}} \right) \end{bmatrix} = \boldsymbol{0}. \tag{4.9}$$

The linearized equations of motion for small values around the point of equilibrium $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{y}_{\mathrm{eq}}$, $\tilde{\boldsymbol{v}} = \boldsymbol{v} - \boldsymbol{v}_{\mathrm{eq}}$ and $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{eq}}$ are

$$\boldsymbol{M}_{\mathrm{lin}} \ddot{\tilde{\boldsymbol{y}}} + \boldsymbol{D}_{\mathrm{lin}} \dot{\tilde{\boldsymbol{y}}} + \boldsymbol{K}_{\mathrm{lin}} \tilde{\boldsymbol{y}} = \boldsymbol{B}_{\mathrm{lin}} \tilde{\boldsymbol{u}} \tag{4.10}$$

$$\boldsymbol{H}_{\mathrm{lin}} \tilde{\boldsymbol{y}} = \boldsymbol{0}, \tag{4.11}$$

with

$$\boldsymbol{M}_{\mathrm{lin}} = \boldsymbol{M}(\boldsymbol{y}_{\mathrm{eq}}) \tag{4.12}$$

$$\boldsymbol{D}_{\mathrm{lin}} = \left(\frac{\partial \boldsymbol{k}}{\partial \dot{\boldsymbol{y}}} - \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{y}}} \right) \Bigg|_{\boldsymbol{y}_{\mathrm{eq}}, \dot{\boldsymbol{y}}_{\mathrm{eq}}} \tag{4.13}$$

$$\boldsymbol{K}_{\mathrm{lin}} = \left(\frac{\partial \boldsymbol{k}}{\partial \boldsymbol{y}} - \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{y}} - \frac{\partial \boldsymbol{B} \boldsymbol{u}_{\mathrm{eq}}}{\partial \boldsymbol{y}} \right) \Bigg|_{\boldsymbol{y}_{\mathrm{eq}}, \dot{\boldsymbol{y}}_{\mathrm{eq}}} \tag{4.14}$$

$$\boldsymbol{B}_{\mathrm{lin}} = \boldsymbol{B}(\boldsymbol{y}_{\mathrm{eq}}) \tag{4.15}$$

$$\boldsymbol{H}_{\mathrm{lin}} = \boldsymbol{H}(\boldsymbol{y}_{\mathrm{eq}}). \tag{4.16}$$

It should be noted that only a single workspace is considered for the analysis and therefore the input is reduced to two forces, therefore $\boldsymbol{u} = \boldsymbol{u}_{\mathrm{red}}$. The linearized equations can be rewritten as

$$\boldsymbol{E}^* \dot{\tilde{\boldsymbol{x}}} = \boldsymbol{A}^* \tilde{\boldsymbol{x}} \tag{4.17}$$

with the augmented state space vector

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \tilde{\boldsymbol{y}} & \dot{\tilde{\boldsymbol{y}}} & \tilde{\boldsymbol{u}} \end{bmatrix}^T \tag{4.18}$$

and the two matrices

$$\boldsymbol{E}^* = \begin{bmatrix} \boldsymbol{I}_f & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{\mathrm{lin}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{4.19}$$

and

$$\boldsymbol{A}^* = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_f & \mathbf{0} \\ -\boldsymbol{K}_{\text{lin}} & -\boldsymbol{D}_{\text{lin}} & \boldsymbol{B}_{\text{lin}} \\ \boldsymbol{H}_{\text{lin}} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{4.20}$$

To obtain information about the stability of the system Eq. (4.17) can be viewed as an eigenvalue problem

$$(\boldsymbol{A}^* - \lambda^* \boldsymbol{E}^*)\boldsymbol{c}^* = \mathbf{0}. \tag{4.21}$$

with λ^* denoting the complex eigenvalues and $\boldsymbol{c}^*$ being the eigenvectors. The eigenvalues λ^* can be either infinitely large or finite. The infinite eigenvalues result from algebraic constraints with infinitely fast motion [GéradinCardona01] while the finite eigenvalues represent the poles of the inverse system, and provide information about the stability of the inverse dynamics [Drücker22].

The locations of the eigenvalues λ^* obtained by Eq. (4.21) are shown in Fig. 4.1 in the case of $N = 2$ segments for parameters given in Tab. 5.1. The length $L_i =$ and mass m_i of both segments are equivalent to the values for $N = 1$ in Tab. 6.1, but divided by 2. A point of equilibrium is computed by choosing an arbitrary force input

$$\boldsymbol{u} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T \tag{4.22}$$

and solving the equations of motion with zero velocity $\boldsymbol{v} = \mathbf{0}$ and zero acceleration $\dot{\boldsymbol{v}} = \mathbf{0}$.

The analyzed system possesses 8 eigenvalues in total, with 4 having a zero real part and infinite imaginary part. The other 4 do have zero real part too and two of them lie at $\text{Im}(\lambda^*) = 4.6502$ while the other two lie at $\text{Im}(\lambda^*) = -4.6502$. Because, no eigenvalue lies in the right half-plane the system's internal dynamics are stable for the case of two segments. This property is assumed to remain for more segments.

4.1.3 Relative Degree

In order to realize trajectory following, the desired output $\widetilde{\boldsymbol{z}}$ has to be continuously differentiable for a control law to exist [Drücker22]. The amount of differentiations required is given by the relative degree r_{rel}. It can be determined

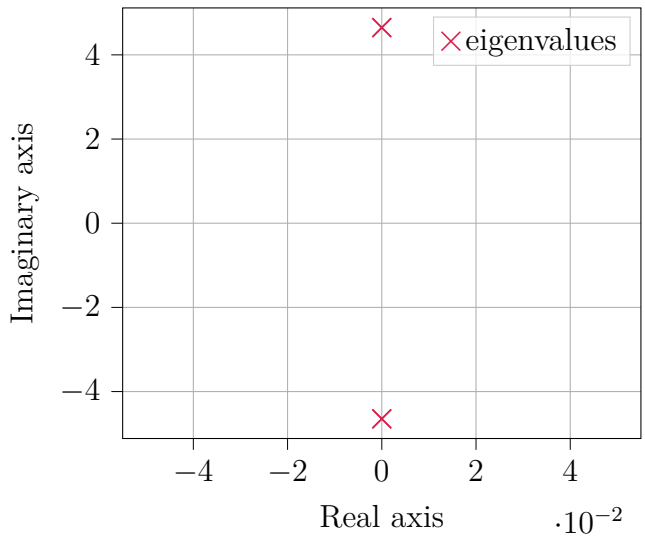


Figure 4.1: The computed eigenvalues for a chosen state of equilibrium of the internal dynamics.

by differentiating the output function $\boldsymbol{z}$ until the input $\boldsymbol{u}$ appears.

The relative degree is determined for a model consisting of two segments. The number of chosen segments is assumed to be independent of the relative degree as the same output trajectory $\boldsymbol{z}$ is chosen for any realization with an arbitrary amount of segments. The output $\boldsymbol{z}$ is defined by Eq. (3.63). Then the output is differentiated twice. With the insertion of Eq. (3.50) for the velocity $\dot{\boldsymbol{v}}$ the acceleration is given by

$$\ddot{\boldsymbol{z}} = \boldsymbol{H}(\boldsymbol{M}^{-1}(\boldsymbol{q} - \boldsymbol{k} + \boldsymbol{B}\boldsymbol{u})) + \dot{\boldsymbol{H}}\boldsymbol{v}. \tag{4.23}$$

Note that Eq. (4.23) is equal to Eq. (4.6), only rearranged. The output had to be differentiated two times until the input $\boldsymbol{u}$ appears and the system has a relative degree of $r_{\text{rel}} = 2$ consequently. In fact, the same parameters in Tab. 5.1 and Tab. 6.1, which were used for the stability analysis in Sect. 4.1.2, were inserted and no input vanished. Therefore any desired trajectory, which the continuum robot should follow later on, needs to be two times differentiable the least.

The input-output relationship, expressed by $\boldsymbol{Y}(\boldsymbol{y})$, confirms all outputs can be directly influenced by the systems input. The matrix is of size 2×2 because $\boldsymbol{H} \in \mathbb{R}^{2\times4}$, $\boldsymbol{M} \in \mathbb{R}^{4\times4}$, $\boldsymbol{B} \in \mathbb{R}^{4\times2}$. $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{M}\boldsymbol{B}$ is invertible as $rank(\boldsymbol{Y}) = 2 = r_{\text{rel}}$. Furthermore, looking from a control theory perspective again, the relative degree

is equal to the difference between finite poles and zeros of the forward system [Drücker22].

4.2 Completing the System

To realize the inverse direction of the system, the servo-constraints have to be chosen in order to achieve an equal amount of differential equations to the number of unknowns. In this work three unknowns in form of cable forces are present. But a realization of servo-constraints for an uneven amount of unknowns is not possible with a non-spatial output $\boldsymbol{z} \in \mathbb{R}^2$. As previously explained, one of the forces is assumed to be zero depending on the actuation area, Fig. 3.8. Therefore, a total number of two servo-constraints is desired. The goal of trajectory following reasons the choice of servo-constraints on positional level. With their addition the overall inverse model can be rewritten in DAE form as

$$\boldsymbol{v} = \dot{\boldsymbol{y}} \tag{4.24}$$

$$\boldsymbol{M}(\boldsymbol{y}, t)\dot{\boldsymbol{v}} + \boldsymbol{k}(\boldsymbol{y}, \boldsymbol{v}, t) = \boldsymbol{q}(\boldsymbol{y}, \boldsymbol{v}, t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{u} \tag{4.25}$$

$$\boldsymbol{s}(\boldsymbol{y}, t) = \boldsymbol{0}. \tag{4.26}$$

The number of equations now equals the number of unknowns and the system can be solved for a state space vector in form of $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{y} & \boldsymbol{v} & \boldsymbol{u} \end{bmatrix}^T$. In fact, with a simple implicit Euler solver the DAE system can be solved by

$$\boldsymbol{0} = -\boldsymbol{x}_{k+1} + \boldsymbol{x}_k + \Delta t \cdot \begin{bmatrix} \boldsymbol{v}_{k+1} \\ \boldsymbol{M}_{k+1}^{-1}(\boldsymbol{q}_{k+1} + \boldsymbol{B}_{k+1}\boldsymbol{u}_{k+1} - \boldsymbol{k}_{k+1}) \end{bmatrix} \tag{4.27}$$

$$\boldsymbol{0} = \boldsymbol{s}(\boldsymbol{y}_{k+1}, t_{k+1}) = \boldsymbol{z}_{k+1} - \widetilde{\boldsymbol{z}}_{k+1} = \boldsymbol{h}(\boldsymbol{y}_{k+1}) - \widetilde{\boldsymbol{z}}_{k+1} \tag{4.28}$$

with the time step $\Delta t = t_{k+1} - t_k$ for the integration and $\widetilde{\boldsymbol{z}}_{k+1}$ denoting the desired trajectory at t_{k+1}, and

$$\boldsymbol{M}_{k+1} = \boldsymbol{M}(\boldsymbol{y}_{k+1}, t_{k+1}) \tag{4.29}$$

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}(\boldsymbol{y}_{k+1}, \boldsymbol{v}_{k+1}, t_{k+1}) \tag{4.30}$$

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}(\boldsymbol{y}_{k+1}) \tag{4.31}$$

$$\boldsymbol{k}_{k+1} = \boldsymbol{k}(\boldsymbol{y}_{k+1}, \boldsymbol{v}_{k+1}, t_{k+1}). \tag{4.32}$$

By adding the servo constraint the solver is determining a solution for the state $\boldsymbol{x}$, that lets the soft robot's tip follow the desired trajectory. The state $\boldsymbol{x}$ includes the system input $\boldsymbol{u}_{k+1}$ and therefore determines the actuation forces, which need to be applied to the soft robot in order to reach the desired tip position.

Chapter 5

Implementation & Setup

This chapter focuses on implementation details and setup of the real robotic system. After a short overview in Sect. 5.2, on how the DAE system is solved, Sect. 5.3 provides the equations for deriving custom trajectories, which will be studied on the real system in Chap. 6. At last, an overview on hardware and characteristics of the real soft robot is provided in Sect. 5.4.

5.1 Software Details

The main simulation environment, visualization and overall framework is implemented in PYTHON. PYTHON is a high-level programming language, offering quick and simple implementation, together with a variety of libraries for scientific application. These are often implemented in fast compiler-based computing languages, such as C and C++. The equations of motion and other system functions, such as calculating the tip position given a state $\boldsymbol{y}$, were derived using MATLAB SYMBOLIC TOOLBOX and exported as C++ functions, to ensure quick evaluation.

5.2 Solving the System

As previously mentioned in Sect. 4.2, the full DAE system Eq. (4.2) can be integrated via the implicit Euler approach. In that regard, first Eq. (3.50) is rearranged and solved for $\ddot{\boldsymbol{y}}$ using the *solve*-function of NUMPY LINEAR ALGEBRA library. The result for $\ddot{\boldsymbol{y}}$ is included in Eq. (4.27) and together with the servo-constraints a solution to $\boldsymbol{x}_{k+1}$ is computed using the *optimize*-library

of PYTHON's SCIPY package.

Especially in the case of six segments, solving to compute $\ddot{\boldsymbol{y}}$ turned out to be challenging, as for certain solver options no solution was found by the library. Starting from an initial guess, which is chosen to be $\boldsymbol{x}_k$, the solution at $k+1$ is calculated, preferably using the *root*-function with the *hybrid*-method [Powell64] of the SCIPY package. However, close to the straight configuration the root finding function struggled to find a solution, even for larger error tolerance. Instead, least squares approach helped finding a solution for the initial straight state, which is why a small area around the robot was defined, where the algorithm would chose *least squares*-function of the SCIPY library. For certain initial deflections it turned out to be beneficial to restrict $\boldsymbol{x}$ to stay withing a certain range, e.g. non-negative values when moving in the positive quadrant. However, starting from the straight configuration remains difficult for arbitrary initial directions, which is why in this work the robot will always move in positive x direction first, before following a trajectory. In fact, often the solver struggled to find an initial solution and consequently fails to solve the system for any following step. In this case, the initial direction for moving out of the straight configuration was changed slightly by moving further into positive quadrant and away from the x-axis, which helped the solver finding a solution. For any future projects it might be beneficial to actuate the robot step by step solving the kinematics only, until a desired starting position is reached.

5.3 Trajectory Generation

Trajectories provide spatial positions and velocities to track for a robot in motion control. In [Mueller19] a trajectory is defined as a function of time returning positional information. A distinction between a purely geometric function definition and a time scaling can be considered to establish a trajectory function [SicilianoEtAl08]. The path only includes information about spacial desired positions and the total trajectory contains the additional timing law. Therefore, generating appropriate trajectories starts with constructing a path omitting time dependency first. Afterwards, the path is joined with time laws, which gives the desired trajectory for the end-effector.

5.3.1 Path Generation

A path is a mathematical function in the form of $\boldsymbol{f}(s)$, with s denoting the parameter, which represents the arc length

$$\boldsymbol{f}(s) = \boldsymbol{p} = \begin{bmatrix} p_x(s) \\ p_y(s) \end{bmatrix}. \tag{5.1}$$

In this work $\boldsymbol{f}(s)$ defines a position vector in the x-y plane of the global frame and is incorporated as $\boldsymbol{f}(s) = \tilde{\boldsymbol{z}}$ in Eq. (4.28). The position of the end-effector is defined as in Eq. (3.63). In the following, three example trajectories are presented, which the soft robot will track.

Straight Line

A very simple path is given by a straight line in space. It is defined by a starting point $\boldsymbol{p}_0$ and an end point $\boldsymbol{p}_e$, connected by a line. The function $\boldsymbol{f}(s)$, describing this path, is defined in [SicilianoEtAl08] by

$$\boldsymbol{f}(s) = \boldsymbol{p}_0 + \frac{s}{s_e}(\boldsymbol{p}_e - \boldsymbol{p}_0) \tag{5.2}$$

with the parameter $s \in [0, s_e]$. The total arc length can be determined by

$$s_e = \|\boldsymbol{p}_e - \boldsymbol{p}_0\|_2 \tag{5.3}$$

but is limited to the radius reachable area of the end-effector. The maximum length of a straight line is equivalent to the diameter of the circular workspace, which is why $s_e = (0, 2\rho]$. Here ρ denotes the radius of the workspace. When s_e is equal to it's upper limit both the starting point $\boldsymbol{p}_0$ and end point $\boldsymbol{p}_e$ lie on the circle defining the workspace.

Circle

Another basic trajectory is a simple circle. Circular trajectories are advantageous paths for robotic manipulators. When the trajectory consists of several rotations around it's central axis, the robot experiences centrifugal forces, which drag the robotic on a circular trajectory in addition to the actuation forces.
A circular path is described by the position of the circle's center $\boldsymbol{c}$ and a point on the radius of the circle $\boldsymbol{p}_0$, marking the starting point of the trajectory. Then the path is described by

$$\boldsymbol{f}(s) = \boldsymbol{c} + r_c \begin{bmatrix} \cos(\frac{s}{r_c}) \\ \sin(\frac{s}{r_c}) \end{bmatrix} \tag{5.4}$$

where r_c denotes the circle's radius. The center of the circle $\boldsymbol{c}$ is chosen as the center of the soft robot's middle axis, here the origin of the global frame

$$\boldsymbol{c} = \mathbf{0}. \tag{5.5}$$

If desired, the circle can be shifted away from the origin, but only as far as to the outer limit of the reachable area of the tip. In other words, the radius r_c, which is given by

$$r_c = \|\boldsymbol{c} - \boldsymbol{p}_0\|_2, \tag{5.6}$$

is limited to the radius ρ of the soft robots workspace $r_c = (0, \rho]$ when Eq. (5.5) holds. The parameter s is limited in the range $s \in [0, s_e]$ again. Now the total arc length s_e is equivalent to the circumference of the circular path

$$s_e = 2\pi r_c. \tag{5.7}$$

In case the soft robot's tip should follow only a circular piece or execute several circular movements, the upper limit s_e can be multiplied by a factor ς. For $0 < \varsigma \leq 1$ the path becomes a circular piece and for $1 > \varsigma$ the path consists of at least one or more circular or partly circular rotations.

Rounded Triangle

The three cable actuation choice motivates a trajectory, which takes the point symmetric bolt hole arrangement into account. An appropriate trajectory design is a smooth triangular path, visualized in Fig. 5.1.

In particular, an outer triangle forms the general path, while polynomials are used to replace the sharp edges by smooth curves. For this work, an isosceles triangle is chosen as the general path. The edges of the rounded triangle are constructed with three linear paths connected by three points $\boldsymbol{p}_A, \boldsymbol{p}_B$ and $\boldsymbol{p}_C$. The curved edges can be represented by polynomials of at least second order. Each polynomial is constrained by the slope of the lines c, which form the outer triangle, and two points, positioned along the triangles edges, marking the transition between the curved corners and straight lines. For example, the purple rounded corner has the transition points $\boldsymbol{p}_{AB}$ and $\boldsymbol{p}_{AC}$. The parameters of a polynomial of second order depend not only on the slope of the straight lines c, but on the positioning of the transition points, too. In other words, the vertex point is determined by the slope and transition points and can not be chosen

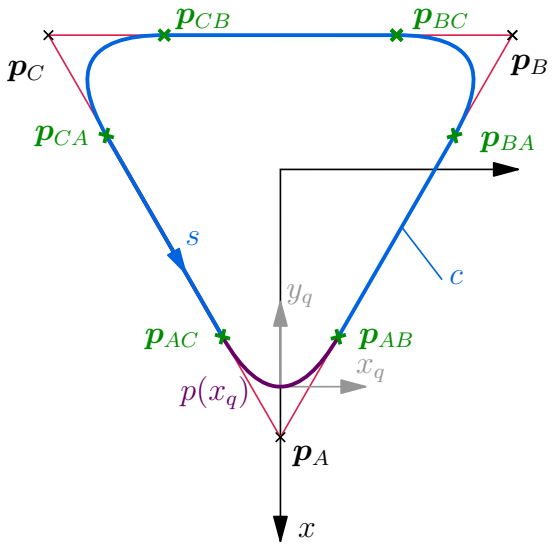


Figure 5.1: Visualization of the triangular trajectory.

freely. A polynomial of higher order can be considered instead, which allows a larger variety of curve designs for the triangles rounded corners. For this work, a quartic function is chosen as

$$p(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \tag{5.8}$$

to model the rounded corners of the triangular trajectory. It is positioned at the origin of the x_q-y_q frame, colored gray in Fig. 5.1. From the triangle's symmetry it is obvious, that the terms of odd order vanish $a_3 = a_1 = 0$, which results in a biquadratic polynomial. The parameters again depend on the slope of the triangle, as well as on the positioning of the transition points. However, the higher order allows a limited shifting of the vertex point. The vertex point can be placed on similar height to the transition points, resulting in a flatter rounded edge or even w type corners. If the vertex point lies closer to the intersection of the lines, the biquadratic function takes a sharper form. The remaining non-zero parameters of the polynomial can be obtained by the constraints

$$p(0) = a_0, \tag{5.9}$$

$$p(x_{AB}) = y_{AB} = a_4x_{AB}^4 + a_2x_{AB}^2 + a_0, \tag{5.10}$$

$$p^{'}(x_{AB}) = c = 4a_4x_{AB}^3 + 2a_2x_{AB} \tag{5.11}$$

$$a_0 = y_{AB} + \alpha(y_A - y_{AB}), \tag{5.12}$$

$$a_2 = \frac{2}{x_{AB}} \left(\frac{y_{AB} - a_0}{x_{AB}} - \frac{c}{4} \right), \tag{5.13}$$

$$a_4 = \left(\frac{1}{y_{AB}} \right)^4 (a_0 - y_{AB} + \frac{c}{2} x_{AB}). \tag{5.14}$$

Here, x_{AB} and y_{AB} are the x and y values of $\boldsymbol{p}_{AB}$. Once the polynomial is determined, the path length of the biquadratic function from $\boldsymbol{p}_{AB}$ and $\boldsymbol{p}_{AC}$ can be calculated. According to [Nystedt21], the path length of a function $f(x)$ is given by

$$s_p = \int_0^{x_p} \sqrt{1 + f'(x)^2}\, dx. \tag{5.15}$$

Then the path length can be obtained by solving

$$s_p = 2 \cdot \int_0^{x_{AB}} \sqrt{1 + 16a_4^2 x^6 + 16a_2 x^4 + 4a_2^2 x^2}\, dx \tag{5.16}$$

using SCIPY library. The path length s_l of the straight edge is calculated with Eq. (5.3). Then the whole path length of the triangular trajectory is given by

$$s_e = 3 \cdot (s_p + s_l). \tag{5.17}$$

The trajectory for the curved corners $j = \{A, B, C\}$ can be expressed by a function

$$\boldsymbol{g}_j(x) = \boldsymbol{R}_j \begin{bmatrix} x \\ a_4 x^4 + a_2 x^2 + a_0 \end{bmatrix} \tag{5.18}$$

defined in the x-y global frame, with $\boldsymbol{R}_j$ denoting the according rotation matrices in the x-y plane

$$\boldsymbol{R}_A = \boldsymbol{I}, \tag{5.19}$$

$$\boldsymbol{R}_B = \begin{bmatrix} \frac{-1}{2} & \frac{-\sqrt{3}}{2}, \\ \frac{\sqrt{3}}{2} & \frac{-1}{2} \end{bmatrix} \tag{5.20}$$

$$\boldsymbol{R}_C = \begin{bmatrix} \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-\sqrt{3}}{2} & \frac{-1}{2} \end{bmatrix}. \tag{5.21}$$

The coordinate x is replaced by $x_j(s)$, which is a linear function determining x given the arc parameter s. The function is defined between the transition points $\boldsymbol{p}_{AB}$ and $\boldsymbol{p}_{AC}$ of the polynomial, given by

$$x_A(s) = x_{AC} + \frac{s}{s_p}(x_{AB} - x_{AC}) \tag{5.22}$$

$$x_B(s) = x_{AC} + \frac{s - (s_p + s_l)}{s_p}(x_{AB} - x_{AC}) \tag{5.23}$$

$$x_C(s) = x_{AC} + \frac{s - 2(s_p + s_l)}{s_p}(x_{AB} - x_{AC}) \tag{5.24}$$

Note, that here the initial polynom at corner A, colored in purple in Fig. 5.1, is rotated with $\boldsymbol{R}_j$ towards the corner j. The three equations differ in the fractions, where previous arc lengths are considered. In fact, the equations can be rewritten by considering the individual transition points of each corner, for example in Eq. (5.23) by using $\boldsymbol{p}_{BA}$ and $\boldsymbol{p}_{BC}$ directly and omitting rotation by $\boldsymbol{R}_B$ in Eq. (5.18). The trajectory for the straight paths is expressed by equations similar to Eq. (5.2), now taking the previous paths into account

$$\boldsymbol{h}_{AB}(s) = \boldsymbol{R}_A \left(\boldsymbol{p}_{AB} + \frac{s - s_p}{s_l}(\boldsymbol{p}_{BA} - \boldsymbol{p}_{AB}) \right), \tag{5.25}$$

$$\boldsymbol{h}_{BC}(s) = \boldsymbol{R}_B \left(\boldsymbol{p}_{AB} + \frac{s - (2s_p + s_l)}{s_l}(\boldsymbol{p}_{BA} - \boldsymbol{p}_{AB}) \right), \tag{5.26}$$

$$\boldsymbol{h}_{CA}(s) = \boldsymbol{R}_C \left(\boldsymbol{p}_{AB} + \frac{s - (3s_p + 2s_l)}{s_l}(\boldsymbol{p}_{BA} - \boldsymbol{p}_{AB}) \right). \tag{5.27}$$

Finally, the overall trajectory can be defined as a piece wise function

$$\boldsymbol{f}(s) = \begin{cases} \boldsymbol{g}_A(s) & \text{for} \quad 0 < s \leq s_p \\ \boldsymbol{h}_{AB}(s) & \text{for} \quad s_p < s \leq s_p + s_l \\ \boldsymbol{g}_B(s) & \text{for} \quad s_p + s_l < s \leq 2s_p + s_l \\ \boldsymbol{h}_{BC}(s) & \text{for} \quad 2s_p + s_l < s \leq 2(s_p + s_l) \\ \boldsymbol{g}_C(s) & \text{for} \quad 2(s_p + s_l) < s \leq 3s_p + 2s_l \\ \boldsymbol{h}_{CA}(s) & \text{for} \quad 3s_p + 2s_l < s \leq s_e \end{cases}. \tag{5.28}$$

It should be noted that the order of the polynomial and linear functions $\boldsymbol{g}$ and $\boldsymbol{h}$ in Eq. (5.28) can be shuffled around. Later, during the experiments in Chap. 6 the desired triangular trajectory follows the linear edges first and then moves along the first round corners. Additionally, the trajectory was mirrored around the x-axis to align with the camera frame.

5.3.2 Adding Time Dependency

As already mentioned in Sect. 4.1.3, the relative degree is $r_{\text{rel}} = 2$. Therefore, the desired trajectory needs to be r_{rel}-times continuously differentiable for a feedforward control law to exist [Drücker22]. Once $\boldsymbol{f}(s)$ is chosen to define the trajectory shape, the path is extended by the time dependency

$$s = s(t). \tag{5.29}$$

Now in order to guarantee a valid feedforward control input the function $s(t)$ needs to be $r_{\text{rel}} = 2$ times differentiable. This can be accomplished by choosing a polynomial of minimum order r_{rel}. However, in order to bound the trajectory to initial conditions as well as ending constraints, the polynomial has to be chosen of higher order [Mueller19]. The trajectory is constrained on position, velocity and acceleration level to accomplish smooth movement. This results in total in $n_c = 6$ constraints for the initial state $t = 0$ and final time at $t = T$. The total of $n_c = 6$ constraints endorses a function $s(t)$ with the same number of unknowns. This can be accomplished by considering a polynomial of fifth order, which possesses variable parameters a_i for $i = 0 \dots 5$

$$s(\tau) = a_5\tau^5 + a_4\tau^4 + a_3\tau^3 + a_2\tau^2 + a_1\tau + a_0 \tag{5.30}$$

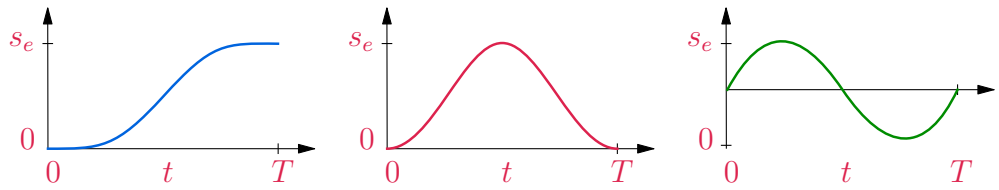
with $\tau = \dfrac{t}{T}$. Then the constraints are set as follows

$$s(0) = 0 \tag{5.31}$$
$$\dot{s}(0) = 0 \tag{5.32}$$
$$\ddot{s}(0) = 0 \tag{5.33}$$
$$s(1) = s_e \tag{5.34}$$
$$\dot{s}(1) = 0 \tag{5.35}$$
$$\ddot{s}(1) = 0. \tag{5.36}$$

With these constraints and the chosen total path length s_e the polynomial becomes

$$s(\tau) = (6\tau^5 + 15\tau^4 + 10\tau^3)s_e. \tag{5.37}$$

Fig. 5.2 visualizes the positional $s(t)$, velocity $\dot{s}(t)$ and acceleration functions $\ddot{s}(t)$ in a given time interval $t \in [0, T]$.



(a) $s(t)$ (b) $\dot{s}(t)$ (c) $\ddot{s}(t)$

Figure 5.2: (a) Positional profile, (b) Velocity profile and (c) acceleration profile.

The positional curve shows how the manipulator moves slowly away from its initial position and smoothly approaches the desired ending point. Concurrently, the manipulator's velocity increases slowly in the beginning, when moving out of it's rest position, and decreases slowly in the end, when reaching it's final destination. The peak velocity is accomplished in between, when the manipulator has built up highest momentum. Finally, the acceleration plot shows how the robot accelerates quickly in the beginning and afterwards drops towards a constant velocity. When the robot nears its final destination, it's acceleration drops again and then the manipulator brakes slowly until the desired end position is reached.

5.3.3 Constant Velocity Profile

For a periodic movement it might be desirable to design a trajectory, which possesses a constant velocity region, instead of continuous non-zero acceleration. [Zelinsky09] describes a straightforward recipe to follow, as well as, covers additional special cases e.g. when the maximum acceleration is not reached. The path's parameter $s(t)$ and its derivative namely the velocity $\dot{s}(t)$, acceleration $\ddot{s}(t)$ and jerk $\dddot{s}(t)$ over time are displayed in Fig. 5.3. The velocity takes the shape of a *Double S*-profile. After a smooth initial phase, characterized by a linear acceleration followed by a linear deceleration phase, the velocity keeps a constant value. Towards the end a mirrored S-shape reduces the speed again.

As seen in Fig. 5.3, three general phases can be distinguished. The first one is the acceleration phase, where the velocity increases. It is further separated into the constant maximum jerk period T_{j1}, followed by the zero-jerk, maximum acceleration period T_a and completed by the minimum jerk period with the same duration T_{j1}. The second is the constant velocity phase. Finally, in the deceleration phase the velocity decreases again. Again, this phase is divided into the same three periods as during the acceleration phase, but in opposite order. The minimum and maximum jerk period is denoted by T_{j2} and the minimum acceleration phase is denoted by T_d.

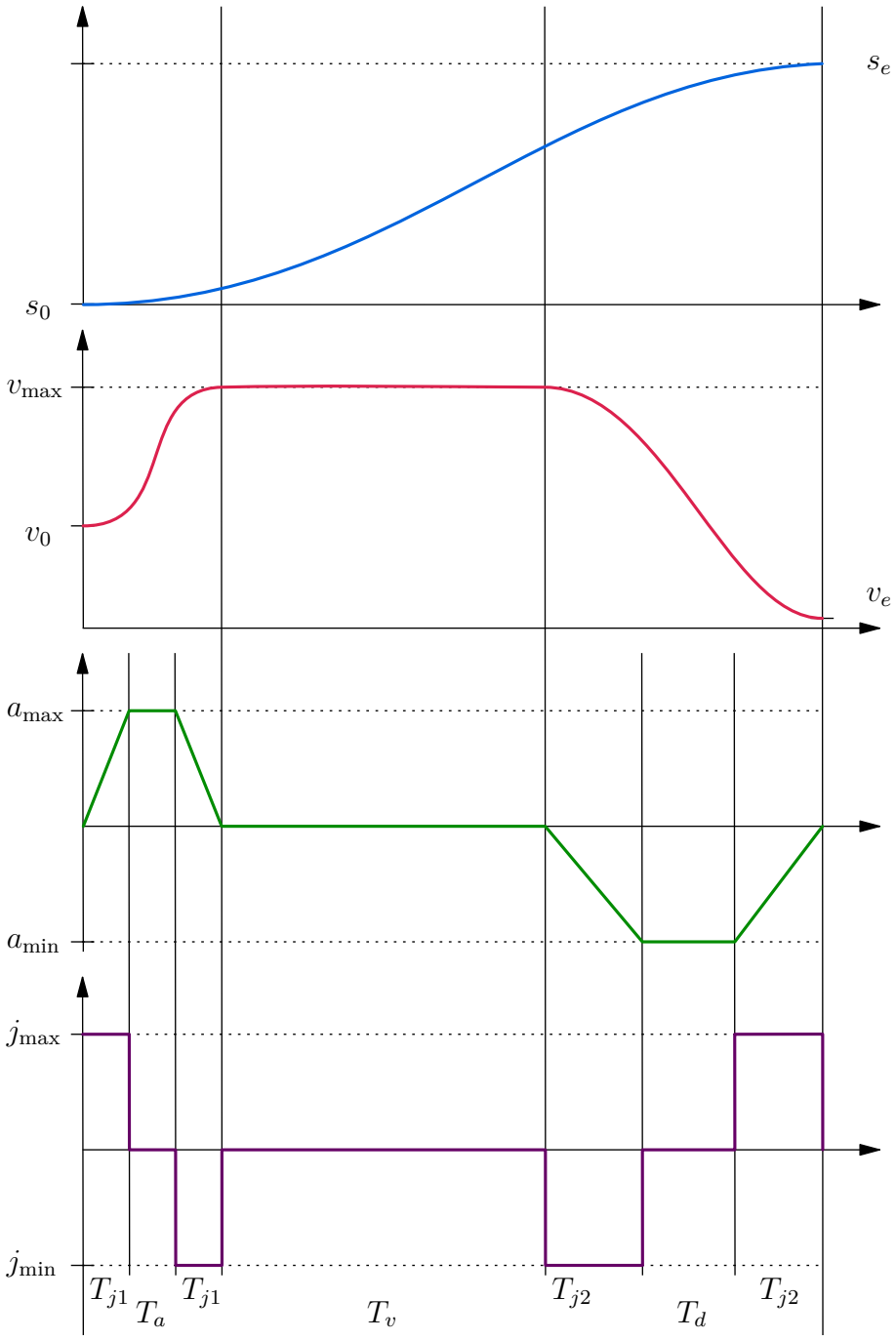


Figure 5.3: *Double S*-profile.

In the following, only a simplified derivation of $s(t)$ for the case of a constant velocity profile will be presented, which is based on certain assumptions and constraints. The complete general approach together with consideration of all possibly occurring special cases can be found in [Zelinsky09]. First symmetric limits are set for $v = \dot{s}(t)$, $a = \ddot{s}(t)$ and $j = \dddot{s}(t)$, resulting in

$$v_{\min} = -v_{\max} \tag{5.38}$$
$$a_{\min} = -a_{\max} \tag{5.39}$$
$$j_{\min} = -j_{\max}. \tag{5.40}$$

Second, zero initial and final accelerations $a(0) = 0$ and $a(1) = 0$, as well as generic zero initial and final velocities $v(0) = 0$ and $v(1) = 0$ are chosen. Further, the s-profile at the end is set to be equal to the initial S-profile but mirrored, which results in equal time segments $T_{j1} = T_{j2} = T_j$ and $T_a = T_d$. At last, $s(0) = s_0 = 0$ is set.

Acceleration phase $t \in [0, T_a]$

The acceleration phase is characterized by a linear profile from the initial zero acceleration to the maximum possible value $a_{\max}$ and then back to zero. Under the assumption of $s_e > s_0$ four possible situations have to be considered. The first case assumes that the maximum velocity is reached $v_{\lim} = v_{\max}$ and a constant velocity segment exists. In this case it is necessary to check if $a_{\max}$ is reached. If this is the case, $v_{\max} \cdot j_{\max} \geq a_{\max}^2$, the periods T_j and T_a are given by

$$T_j = \frac{a_{\max}}{j_{\max}} \tag{5.41}$$
$$T_a = T_j + \frac{v_{\max}}{a_{\max}}. \tag{5.42}$$

If the maximum acceleration $a_{\max}$ is not reached $v_{\max} \cdot j_{\max} < a_{\max}^2$, the periods T_j and T_a are

$$T_j = \sqrt{\frac{v_{\max}}{j_{\max}}} \tag{5.43}$$
$$T_a = 2T_j. \tag{5.44}$$

Then, the duration of the constant velocity phase is given by

$$T_v = \frac{s_e - s_0}{v_{\max}} - T_a. \tag{5.45}$$

For the period with constant velocity $T_v > 0$ the maximum velocity is reached $v_{\lim} = v_{\max}$, otherwise the case $v_{\lim} < v_{\max}$ needs to be considered and the duration becomes $T_v = 0$. Again, it is necessary to check if the maximum acceleration $a_{\max}$ is reached. For $(s_e - s_0) \geq 2\frac{a_{\max}^3}{j_{\max}^2}$, T_j and T_a are obtained by

$$T_j = \frac{a_{\max}}{j_{\max}} \tag{5.46}$$

$$T_a = \frac{T_j}{2} + \sqrt{\left(\frac{T_j}{2}\right)^2 + \frac{s_e - s_0}{a_{\max}}}. \tag{5.47}$$

In case of $(s_e - s_0) < 2\frac{a_{\max}^3}{j_{\max}^2}$, T_j and T_a are given by

$$T_j = \sqrt[3]{\frac{s_e - s_0}{2j_{\max}}} \tag{5.48}$$

$$T_a = 2T_j. \tag{5.49}$$

Maximum Velocity Phase $t \in [T_a, T_a + T_v]$

The maximum velocity phase is present only for a constant velocity period $T_v > 0$. Then the duration T_v is computed by Eq. (5.45) and the robot moves with the limited velocity being equal to the maximum velocity $v_{\lim} = v_{\max}$. If the maximum velocity is not reached, $T_v = 0$ and the velocity profile takes a similar shape to Fig. 5.2, with $v_{\lim}$ being the highest velocity value.

Deceleration Phase $t \in [T_a + T_v, T]$

Because of the previously introduced constraints of zero initial and final velocity and the symmetric limits the second s-profile during deceleration phase takes the same shape as the s-profile during the acceleration process. This results in $T_d = T_a$.

Path Function

Finally, $s(t)$ and its derivatives can be constructed as piece-wise functions. With the determined time segments the maximum and minimum accelerations and maximum velocity (for the case of $v_{\text{lim}} < v_{\max}$) can be calculated with

$$a_{\text{lim},a} = j_{\max} T_j = -a_{\text{lim},d} \tag{5.50}$$

$$v_{\text{lim}} = (T_a - T_j)\, a_{\text{lim}} \tag{5.51}$$

Then the parameter $s(t)$ is defined as

$$s(t) = \begin{cases} s_0 + v_0 t + j_{\max} \frac{t^3}{6} & \text{for } t \in [0, T_j] \\ s_0 + v_0 t + \frac{a_{\text{lim},a}}{6} T_\beta & \text{for } t \in [T_j, T_a - T_j] \\ s_0 + v_{\text{lim}_0} \frac{T_a}{2} - v_{\text{lim}} T_\alpha - j_{\min} \frac{T_\alpha^3}{6} & \text{for } t \in [T_a - T_j, T_a] \\ s_0 + -v_{\text{lim}} T_1 \left(v_{\text{lim}} + v_0\right) \frac{T_a}{2} & \text{for } t \in [T_a, T_a + T_v] \\ s_e - v_{\text{lim}_1} \frac{T_d}{2} + v_{\text{lim}} T_\gamma - j_{\max} \frac{T_\gamma^3}{6} & \text{for } t \in [T - T_d, T - T_d + T_j] \\ s_e - v_{\text{lim}_1} \frac{T_d}{2} + v_{\text{lim}} T_\gamma + \frac{a_{\text{lim},d}}{6} T_\delta & \text{for } t \in [T - T_d - T_j, T - T_j] \\ s_e + v_1 (T - t) - j_{\max} \frac{(T-t)^3}{6} & \text{for } t \in [T - T_j, T] \end{cases} . \tag{5.52}$$

with

$$T_\alpha = T_a - t \tag{5.53}$$

$$T_\beta = 3t^2 - 3T_j t + T_j^2 \tag{5.54}$$

$$T_\gamma = t - T + T_d \tag{5.55}$$

$$T_\delta = 3t^2 - 3T_j t + T_j^2 \tag{5.56}$$

$$v_{\text{lim}_0} = v_{\text{lim}} + v_0 \tag{5.57}$$

$$v_{\text{lim}_1} = v_{\text{lim}} + v_1 \tag{5.58}$$

Here, the function for the path $s(t)$ is provided only. The remaining functions for $v(t)$, $a(t)$ and $j(t)$ can be found in [Zelinsky09].

5.3.4 Shifting Trajectories

In order to analyze the soft robot's capability of following chosen trajectories, it is favorable to be able to generate trajectories with an offset towards the soft robot's center, or, in case of the triangular trajectory, rotate the overall trajectory around the robot's longitudinal axis. For this purpose, each of the functions $\boldsymbol{f}(s)$

describing the trajectory can be multiplied by a rotation matrix $\boldsymbol{R}_\psi$ and shifted in the x-y plane by an adding an offset $\boldsymbol{p}_{\text{off}}$

$$\bar{\boldsymbol{f}}(s) = \boldsymbol{p}_{\text{off}} + \boldsymbol{R}_\psi \boldsymbol{f}(s). \tag{5.59}$$

5.4 The Real Soft Robot

The soft flexible rod is the main body of the overall system. It has been manufactured by molding HT 45-silicone into beforehand 3D-printed molds. To enable bending ability, the rod possesses a caterpillar-like shape, as seen in Fig. 5.4

The convexly shaped bulges build the mass elements, while the concave formed gaps sustain defined bending of the rod. A total of six bolt holes are integrated into each of the mass elements, which allow the guidance of cable for actuation. As described in Sect. 3.4, three cables are used to control the robot in this work. Additional three cables can be considered to allow more complex shapes. The cables have to be attached to the robots tip, which in this work is a 3D-printed cube, see Sect. 5.6. The flexible rod itself is placed into a 3D-printed socket, which is fixed onto a ground plate, where the lose cable ends are guided through in order to be pulled by a suitable actuation choice.
Measured sizes such as lengths and masses as well as precalculated quantities are given Tab. 5.1. All of these values can be determined through calculation or precise measurement to sufficient accuracy and are independent of any other quantities. For example, the Young's modulus E can be approximated by

$$E = \frac{0.0981(56 + 7.62336 \cdot H)}{0.137505(254 - 2.54 \cdot H)} = 2.04\,\text{MPA} \tag{5.60}$$

according to [Gent58] for small non-linear deformation. Here, $H = 45\,\text{ShA}$ represents the Shore-hardness of the HT 45-silicone.

5.5 Actuation System

As previously mentioned, the softrobot is actuated by attached wires, whose loose ends stick out of the socket. Those ends are wrapped around 3D-printed reels, which in turn are mounted on small *Hitec HS311* motors. The *Hitec HS311* motors are limited in their actuation power resulting in a limited diameter for the reels. Additionally, the motors do not allow continuous rotation or idle and instead a rotation range of 180° is possible. To enable sufficient bending for

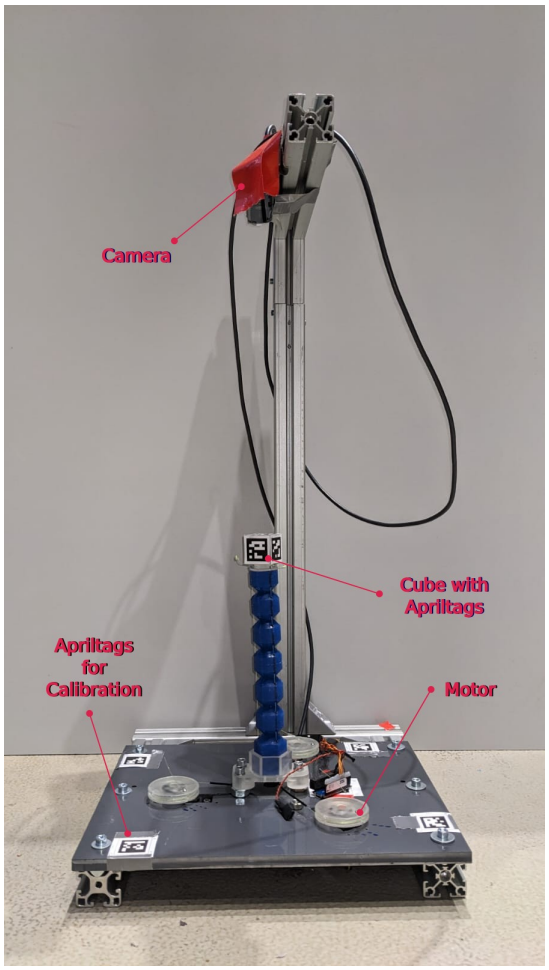


Figure 5.4: The complete robot system, with the soft robot and APRILTAGS-cube in the middle, camera at the top and motors for actuation mounted around.

Table 5.1: Measures held constant for any segment realization.

Quantity	Symbol	Value	Unit
segment radius	r	15	[mm]
bolt hole radius	r_{holes}	12.5	[mm]
bolt hole length	l_{holes}	14	[mm]
Young's modulus	E	2.04	[MPa]
Second moment of inertia	$I_{\text{loc},xx}$	3.976×10^{-8}	[kgm^2]
reel diameter	d_{reel}	46	[mm]
APRILTAGS cube size	-	0.03	[mm]
APRILTAGS cube mass	-	0.0172	[kg]

studying the robot's dynamic behavior, the reels should possess a sufficiently large diameter on the one hand, but are limited in size due to the motors limited actuation power. Therefore, a fitting diameter was determined, listed with the other dimensions in table Tab. 5.1.
The motors are further connected to a ARDUINO NANO-board, that serves as an interface between the motors and a connected computer. The connected computer allows fast computation power to solve the equations of motion, if desirable even in real-time. However, in this work the inputs are precalculated for simplicity. Then pulse-width modulation (PMW) is used to control the motors to rotate about the calculated angles.

5.6 Tracking Setup

To validate the soft robot's ability to follow a desired trajectory in space a webcam is placed right above the robot's tip. The tracking is accomplished by gluing pictures of APRILTAGS to the sides of the tip's cube, which are recognized by the camera system. The APRILTAGS resemble simplified QR-codes, which are a common and popular method for visual tracking in research development, as conventional camera systems are capable of detecting the tags [Olson11]. For calibration purposes, additional APRILTAGS are positioned on the ground plate of the physical system. These are necessary to determine the position of the camera in space. Together with the OPENCV library the spacial position of the APRILTAGS can be measured and the cube's center position can be calculated in the 2D projection camera frame [Itseez15]. This accomplishes validation and evaluation of the model as well as tuning of uncertain parameters of the system.

Chapter 6

Experimental Results

This chapter focuses on experimental results with the real soft robot. The first part of this chapter is dedicated to identification of unknown parameters and their influence on the robots capability of reaching a desired point in the task space. Sect. 6.1 presents the according parameters and introduces different tuning strategies to determine appropriate values. The second part of this chapter then studies trajectory following under different system settings. The desired trajectories and according system variables influencing the soft robots dynamic behavior are examined in Sect. 6.2.

6.1 Parameter Identification

All parameters used in modeling soft robot are either determined by measurements, computation, or can be chosen by the user. Those quantities, which remain constant and do not depend on a specific use case, are listed in Tab. 5.1. Parameters, which are dependent on the number of chosen segments for the PCC method, are displayed in Tab. 6.1. It should be noted, that for the last segment the additional mass of the Apriltags-cube is added on top. The weight of the cube is given in Tab. 5.1.

Table 6.1: Parameter dependent on the number of segments N.

Quantity	Symbol	Value ($N = 1$)	Value ($N = 3$)	Value ($N = 6$)	Unit
segment length	L_i	190	60	30	[mm]
segment mass	m_i	0.11693	0.03648	0.01824	[kg]

User-chosen inputs include all parameters, which are dependent on the desired use case. These include all quantities that describe the trajectory or the dynamic behavior of the soft robot. The values of these parameters can be freely selected within certain limits. For example, the robot can not follow a trajectory that is out of its reach or move with any arbitrary speed. These parameters will be examined further in Sect. 6.2, when their influence on following a desired trajectory is studied. The parameters are listed in Tab. 6.6.

Finally, there are parameters, which are unknown and need to be identified. These parameters are either unmeasureable or very sensitive to inaccuracies in manufacturing or/and montage. In this work, these parameters are related to the actuation input, denoted by the change in cable lengths Δl_F and Δl_K. As already explained in Sect. 3.4, a virtual linear spring with stiffness coefficient c_q is modeled to take the elongation of each actuation cable into account. A value for each c_q can be obtained through appropriate experiments prior to deployment of the system, but any manufacturing and montage inaccuracies are neglected then. Similar to each change in cable length $\Delta l_{F,q}$, every $\Delta l_{K,q}$ is affected by the system defects and inaccuracy in modeling, which is why another parameter b_q is introduced to take them into account. All together the parameterized total cable length input to the system is given by

$$\Delta l_q = \frac{F_{\mathrm{cab},q}}{c_q} + b_q \Delta l_{K,q}. \tag{6.1}$$

With the unknown parameters introduced, the question arises if they are independent of any other quantities and how to find an appropriate representation. In the most simple case, both parameters c_q and b_q are independent of any other parameter or quantity, which would result in constant factors for all 6 parameters. If this is not the case, their values need to be represented by an appropriate function, which takes the dependent quantity into account.

6.1.1 Impact of the Coefficients $\Delta l_{F,q}$ and $\Delta l_{K,q}$

Before tuning the parameters c_q and b_q, their impact on the system has to be examined. To later determine appropriate values for following desired trajectories it is essential to understand how their choice will influence the outcome. Because of that, first both factors will be studied individually on a circular trajectory with $r_c = 5\,\mathrm{cm}$, to obtain a better understanding of their impact. The time profile is chosen as defined in Eq. (5.37). Further properties of the chosen trajectory are given in Tab. 6.2.

Table 6.2: Trajectories for parameter identification.

Shape -	Size r_c [cm]	Duration (total) t_{dur} [s]	Duration (initial) t_{init} [s]	Num. of laps n_{lap}
Circle	2	20	3	1
Circle	3	20	3	1
Circle	4	20	3	1
Circle	5	20	3	1
Circle	6	20	3	1
Circle	8	20	3	1
Circle	10	20	3	1
Circle	12	20	3	1

Impact of Stiffness Coefficient c_q

For determining the impact of the stiffness c_q on the outcome, only the force input will be considered, by setting

$$b_q = 0 \tag{6.2}$$

for all actuators $q = 1 \dots 3$. The resulting effects are shown Fig. 6.1 with the according stiffness c_q of the virtual spring given in the first column of Tab. 6.3. It shows the measured tip position $\boldsymbol{z}^{\text{meas}}$ in blue, together with the calculated tip position $\boldsymbol{z}^{calc}$ in green, obtained by evaluating Eq. (3.63), and the desired trajectory $\widetilde{\boldsymbol{z}}$ in black. Note, that the calculated tip positions are very accurate and therefore conceal the desired trajectory. With only force actuation the robot approximates the circular trajectory by a hexagonal. This is due to the actuation design described in Sect. 3.4. Only two motors are pulling the robot at the same time while the third motor is resting. A constant load is acting on the respective cable while the robot tries to bend away from the jammed motor. Consequently, the robot is not able to move on a circular path. This is where additional actuation with $\Delta l_{K,q}$ will help the robot.

Impact of Kinematic Cable Length $\Delta l_{K,q}$

The addition of the kinematic cable length $\Delta l_{K,q}$ allows the robot to accomplish circular motion in the x-y plane. The non-actuating motor then doesn't rest, but instead moves along and does not constrain the robot in its motion. If only the length $\Delta l_{K,q}$ is considered for actuation, the robot is able to perform circular motion, however, the elongation of the cables during calculation is not considered, which may have large influence on the performance of the system. In Fig. 6.2,

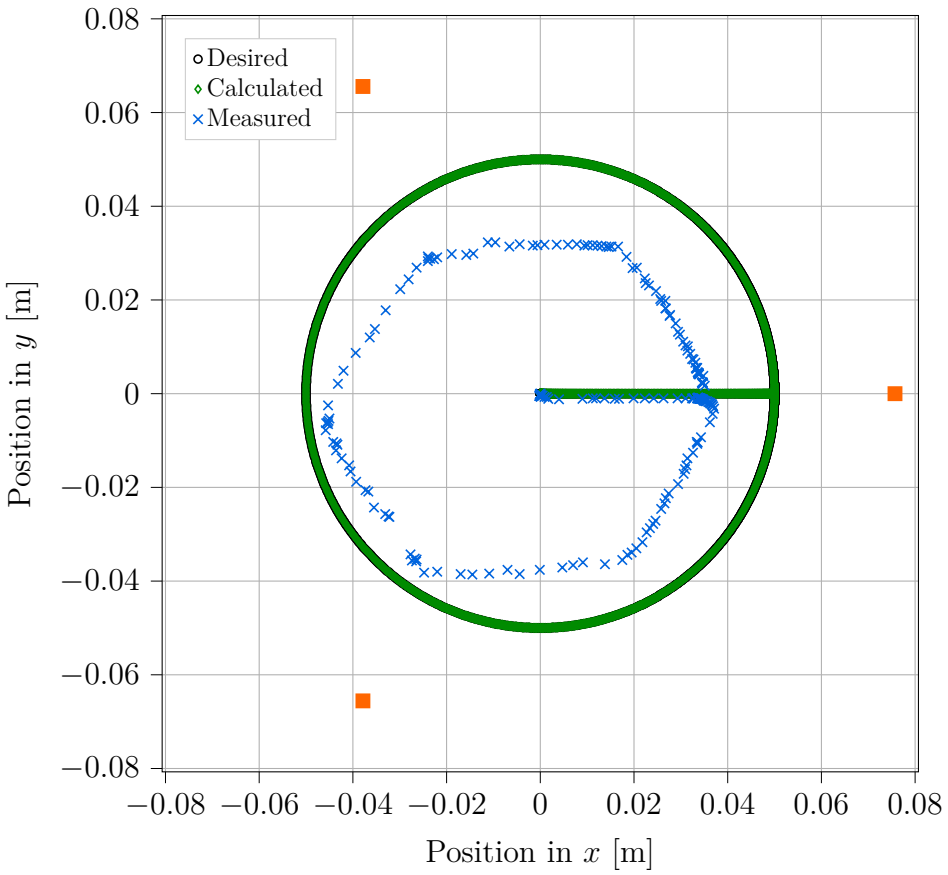


Figure 6.1: Recorded Trajectory (blue) when only coefficient $\Delta l_{F,q}$ is used as input with desired trajectory (green). The orange squares represent the motors.

only the change in length $\Delta l_{K,q}$ was utilized for following a circular trajectory with $r_c = 0.05$m. The values for the parameters b_q and c_q are listed in the second column of Tab. 6.3. Note, that all parameters c_q for $q = 1 \dots 3$ have very large values. The round shape is apparent but overshoots and offsets are clearly noticeable in the driven path. For example, on the left side the robot exceeds the desired trajectory. At some points the robot does not manage to stay on the trajectory and cuts short. A larger parameter b_q would increase the pulling of the active motors, but at the same time loosen further when being not active. Then less power is required by the active motors to pull the robot again. This is where additional actuation with the length $\Delta l_{F,q}$ will be beneficial, as it allows the motor to pull stronger when actuating and still loosen for the same amount when not active.

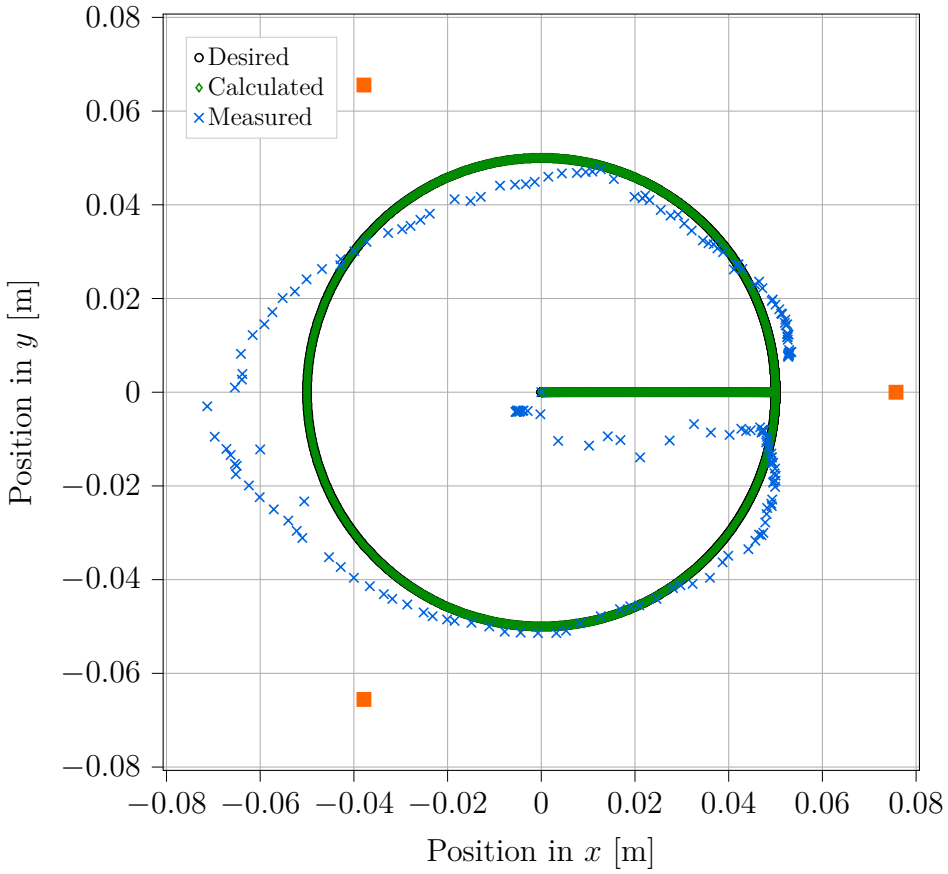


Figure 6.2: Recorded Trajectory when only coefficient $\Delta l_{K,q}$ is used as input.

Table 6.3: Tuned input parameters for radius $r_c = 5\,\text{cm}$.

Parameter	Impact of c_q	Impact of b_q	Approach 1
c_1 in $\frac{\text{N}}{\text{m}}$	400	$2{\times}10^5$	1350
c_2 in $\frac{\text{N}}{\text{m}}$	350	$2{\times}10^5$	1100
c_3 in $\frac{\text{N}}{\text{m}}$	500	$2{\times}10^5$	1050
b_1	0	1.7	1.05
b_2	0	1.4	1.15
b_3	0	1.3	1.2

6.1.2 Tuning of the Parameters c_q and b_q

With the resulting effects of the coefficients $\Delta l_{F,q}$ and $\Delta l_{K,q}$ illustrated, the proportion of both during actuation needs to be determined. This is achieved by finding appropriate parameters c_q and b_q, which will force the robot to follow the desired trajectory. Because of the possible dependency on other parameters and system settings, a fitting representation needs to be determined. For this purpose three strategies are considered:

- constant values for all parameters,
- scaling of the parameters by considering the distance to the origin,
- identifying appropriate values for certain distances and designing a function by curve fitting.

Approach 1: Constant Parameters

The most simple approach is given by modeling both parameters as constant factors, which reinforce or reduce the influence of the coefficients $\Delta l_{F,q}$ and $\Delta l_{K,q}$. All parameters are tuned until the robot manages to approximate the circular movement and are listed in Tab. 6.3.

The first trajectory of circular shape with radius $r_c = 5\,\text{cm}$, given in Tab. 6.2, is considered. Fig. 6.3 shows the result after the parameters were tuned to follow the desired trajectory. A series of 10 experiments was taken and the result with the lowest average error in distance

$$n_{\text{best}} = \arg\min \frac{1}{T} \sum_{t=0}^{T} \|\widetilde{\boldsymbol{z}}_t - \boldsymbol{z}_t^{\text{meas}}\| \tag{6.3}$$

was chosen. Here, t marks the time where the measurement $\boldsymbol{z}_t^{\text{meas}}$ was taken. The corresponding desired position $\widetilde{\boldsymbol{z}}_t$ is obtained by linear interpolation. It should be noted that measurements taken for the parameter identification include a time delay, which stays consistent over a series of measurements. The delay differs only minimally between different trajectories but has been eliminated. The reason for this delay in general originates from the visual measurements, as the pose estimation and transformations for tracking the center of the Apriltags cube was performed in real-time. During the experiments on dynamic behavior only the Apriltags themselves were tracked in real-time, while the computation of the cube's center is performed after the run finishes. Thus, the time delay

becomes smaller in later experiments, but still remains.

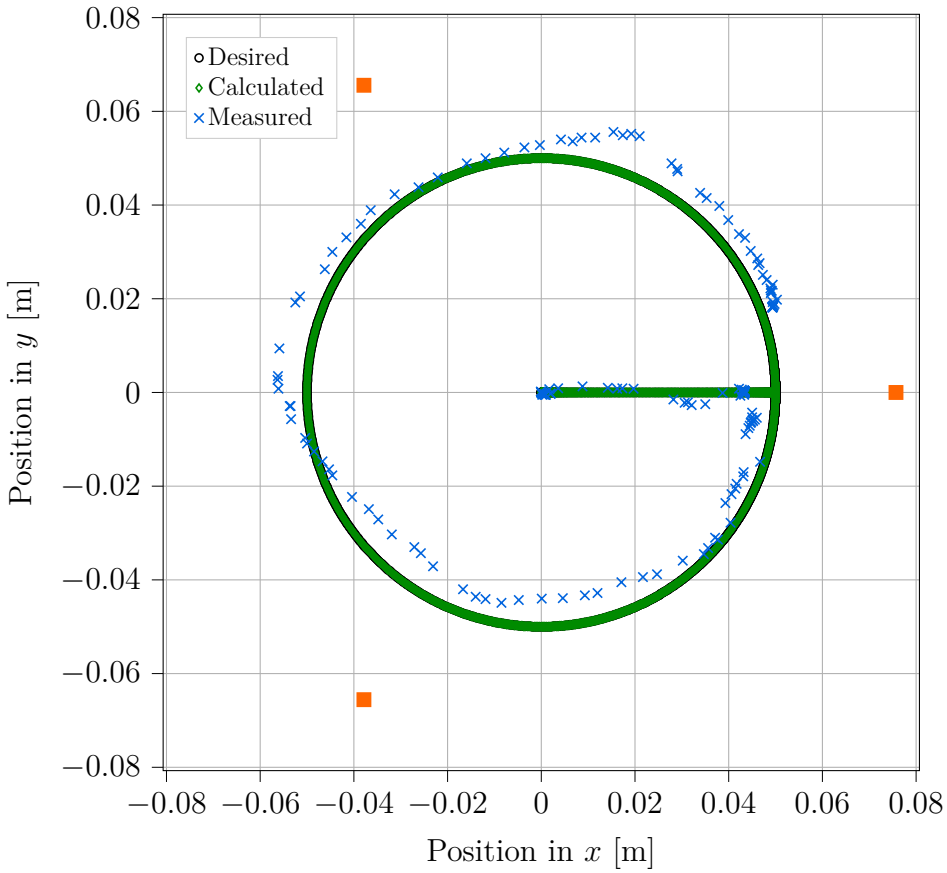


Figure 6.3: Recorded Trajectory for a circle of radius $r_c = 5\,\text{cm}$ with parameters for approach 1 given in Tab. 6.3.

It is clearly noticeable that the robot is capable of following the circular trajectory to a certain extend, but deviates from the desired path at a few spots. Those deviations occur either consistently or every once in a while. Permanent offsets arise due to not ideally tuned parameters. In contrast, varying deviations appear due to minor noise in each run, for example due to random friction.

The high variation in motion makes the tuning challenging, as often slight overshoots and offsets arise during motion. Further, all 6 parameters do not only affect the local area around the according motor, but can influence the behavior of the robot everywhere. For example, a constant overshoot appears at the left side, between motor 2 and 3. Here either the parameters c_2 and c_3 can be increased, b_2 and b_3 decreased or b_1 reduced only. Changing the parameters c_1 and c_3 results in less strain induced by the motors 2 and 3. However, in the

area close to motor 2, where the robot cuts short, the deviation would increase further. Decreasing the parameters b_2 and b_3 causes less tension by the motors 2 and 3, too. But this change effects the performance on the opposite site, too, where the robot is predominantly actuated by motor 1. The motors 2 and 3 restrain the motion and the parameter c_1 has to be decreased further because otherwise the robot will fail to reach the desired radius. Alternatively, lowering the parameter b_1 only will give the robot less freedom in its movement, but will lead to sharper edges and less curvature motion. Additionally, a change of the parameter b_1 will affect the robot's performance everywhere.

In general, reducing the parameters c_q allows the according motor to pull the robot closer to it, without affecting the non-active actuation area. This results in sharper edges for the trajectory and a hexagon shape evolves, see Fig. 6.1. But general constant deviation can be prevented, as the offset around motor 2 in Fig. 6.3. Contrarily, larger parameters c_q allow more curvy motion but the elongation of the cables is considered less, which causes a more deviation over a greater span, as the offset around motor 2. Larger parameters b_q support the execution of non-linear motion. When a motor is not-active it adapts to the movement forced by the pulling motor(s) by reducing the tension on it's own cable. Therefore an increase of b_q can reduce the sharp peaks, induced by the coefficient $\Delta l_{F,q}$, but at the same time allows less tension from the non-active motor, which can lead to more overshoots and offsets everywhere, as seen on the left in Fig. 6.2.

Tuning the parameters c_q and b_q by hand is challenging because of their contrary behavior as both values influence each others impact and because of the reoccurring minor disruptive factors, such as friction or noise in measurements. However, as the results in Fig. 6.3 show, it is possible to tune these parameters to certain extend to let the robot follow a chosen trajectory. With the tuned parameters present, other circles with larger and smaller radii are considered. The according trajectories are listed in Tab. 6.2. For each trajectory only the radius was changed and all of the other parameters are kept the same. Two errors are defined to compare the results. The maximum and average error over time are given as

$$e_{\text{avg}} = \text{mean}\left(\frac{\|\widetilde{\boldsymbol{z}}_t - \boldsymbol{z}_t^{\text{meas}}\|}{r_c}\right) \tag{6.4}$$

$$e_{\text{max}} = \max\left(\frac{\|\widetilde{\boldsymbol{z}}_t - \boldsymbol{z}_t^{\text{meas}}\|}{r_c}\right). \tag{6.5}$$

It is important to note, that both errors are scaled with the radius in order to ensure a proportionate comparison for the error values. Therefore the errors

have no unit. The minimum values of the two errors e_{avg} and $e_{\max}$ of a series of 10 measurements are shown in Tab. 6.4.

Table 6.4: Minimum average and maximum error for different circular trajectories.

r_c in m	0.02	0.03	0.04	0.05	0.06	0.08	0.1
e_{avg}	0.4053	0.2385	0.1308	0.1558	0.1525	0.1356	0.1447
$e_{\max}$	0.6767	0.3858	0.3405	0.3237	0.3166	0.299	0.3325

The results show that both errors increase for significant smaller radii, but remain similar for larger trajectories. Additionally, Fig. 6.4 and Fig. 6.5 show the according trajectories to the given values in Tab. 6.4.

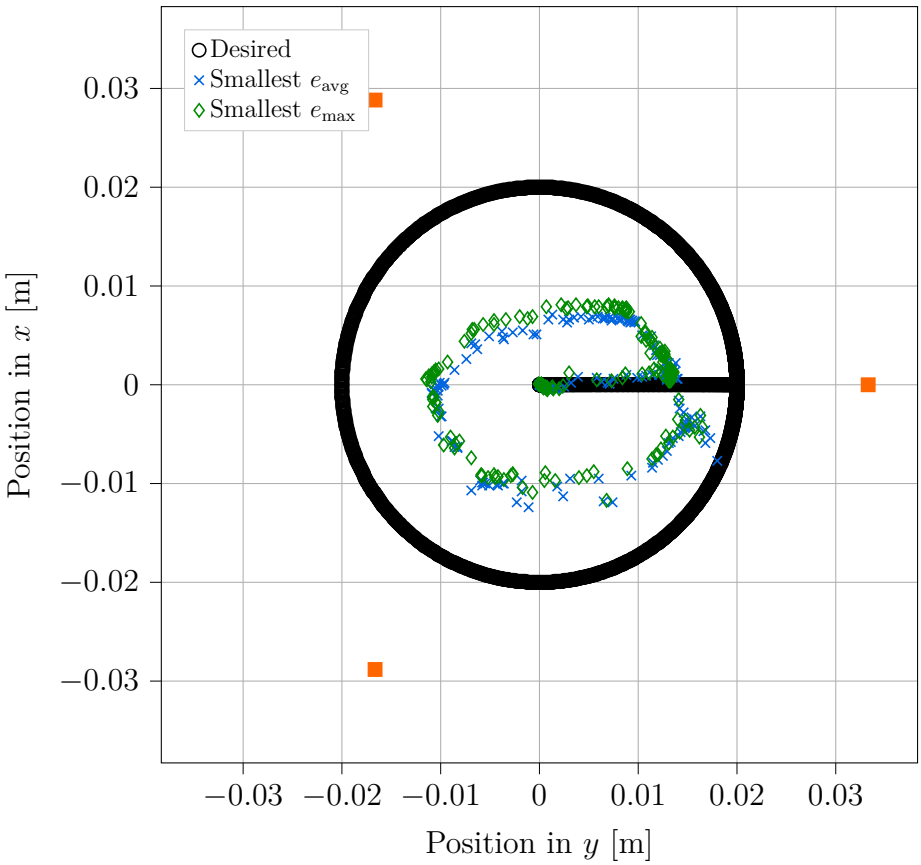


Figure 6.4: Recorded trajectory error for a circle of radius $r_c = 2\,\text{cm}$ with the parameters for approach 1 given in Tab. 6.3.

It is clearly visible that the robot does follow the trajectories worse compared to the circular path with radius $r_c = 5\,\text{cm}$. For a larger circle, the robot manages to follow the path approximately, but with a general offset. This is not the case for a circle with significant smaller radius, where the robot does not reach the desired path at all. Here, less tension is required for smaller deflections compared to trajectories, where the robot is bent stronger to reach the desired point in space. The aforementioned inaccuracies in manufacturing and mounting as well as the effects of friction between the cables and reels have a greater impact on the system and elongation of cables are clearly visible. These disturbances have lesser influence for larger sizes, where the actuation strain has a far stronger impact in comparison.

In general, the robot's tip overshoots for circles with a significant larger radius. Contrarily for trajectories close to the robot's center, the end-effector does not reach the desired path and instead lingers around the origin. The results show, that especially for smaller trajectories the constant parameter approach is not

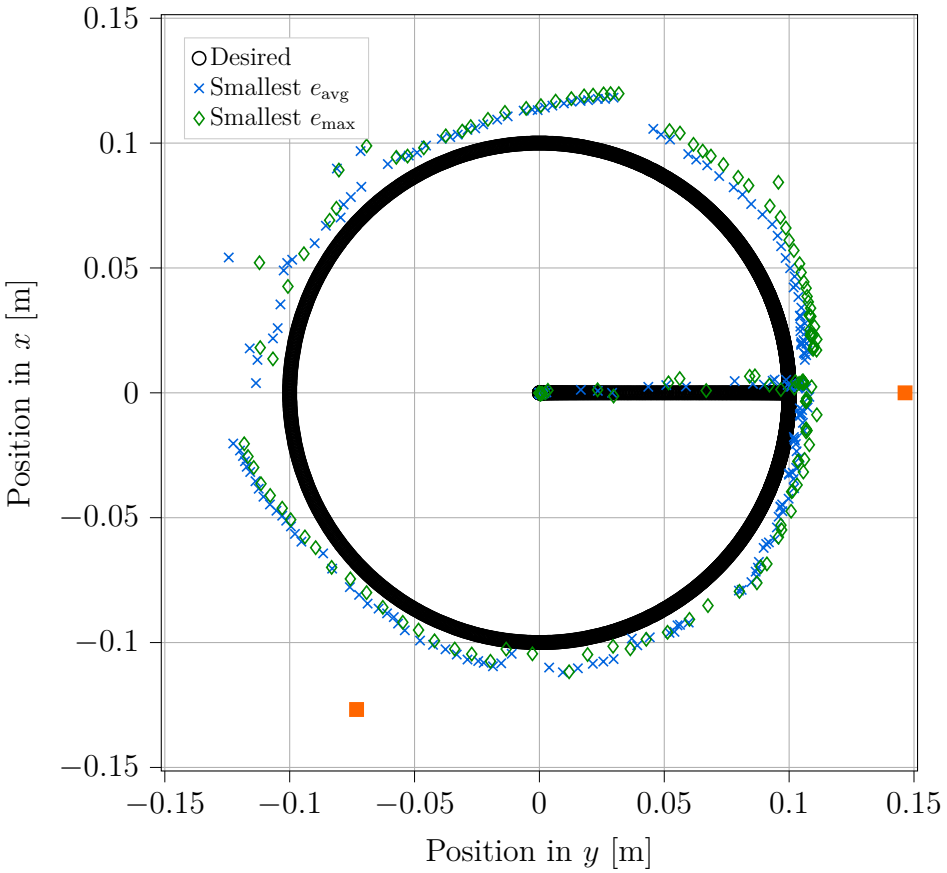


Figure 6.5: Recorded trajectory errors for a circle of radius $r_c = 10\,\text{cm}$ with parameters for approach 1 given in Tab. 6.3.

sufficient. Likewise, tracking of trajectories with larger size can be improved as a constant offset is noticeable.

Approach 2: Linear Parameters

The first approach with constant parameters is not sufficient for tracking trajectories, in particular for paths close to the robot's center. Alternatively to the first approach, the parameters can be split into a constant factor multiplied by the radius of the circular path. Then a parameter is scaled according to the radius. For any arbitrary trajectory, a function can be defined to represent each parameter, which takes in the distance between the origin and the current tip position in the x-y task space and returns the scaled parameter.

The results for constant parameters show that the robot struggles to reach the trajectory for circles with a radius $r_c < 5\,\text{cm}$ and exceeds for trajectories with a radius $r_c > 5\,\text{cm}$. This motivates five ideas for considering the size of the trajectory.

In the previous approach optimal values for the parameters c_q and b_q for a circular trajectory with radius $r_c = 5\,\text{cm}$ were found experimentally. These values can be separated into the product of a constant factor and the radius r_c of the desired circular path. Then the parameter c_q is given by

$$c_q = r_c \cdot c_q^{'} \tag{6.6}$$

and the parameter b_q by

$$b_q = \frac{b_q^{'}}{r_c}. \tag{6.7}$$

Here, the two values $c_q^{'}$ and $b_q^{'}$ represent the new constant factors. They are calculated from the results with a circle of $r_c = 5\,\text{cm}$. The constant $c_q^{'}$ is given by rearranging Eq. (6.6) to

$$c_q^{'} = \frac{c_q^5}{0.05}. \tag{6.8}$$

and the value for the constant $b_q^{'}$ is obtained by rearranging Eq. (6.7) to

$$b_q^{'} = b_q^5 \cdot 0.05. \tag{6.9}$$

The inclusion of r_c for scaling the parameters can happen in five different ways. First, only one of the parameters is scaled by considering only Eq. (6.6) or Eq. (6.7) for computing the total change in cable length in Eq. (6.1). Second both parameters are scaled with Eq. (6.6) and Eq. (6.7). Third, if the effects of scaling for one of the two parameters is way larger than the other, the inverse of r_c can be considered, resulting in scaling according to

$$c_q = \frac{c_q^{''}}{r_c}, \tag{6.10}$$

$$c_q^{''} = c_q^5 \cdot 0.05, \tag{6.11}$$

for the spring stiffness and

$$b_q = b_q^{''} \cdot r_c, \tag{6.12}$$

$$b_q^{''} = \frac{b_q^5}{0.05}, \tag{6.13}$$

for the kinematic cable length input. The resulting change in cable length is then given for the five cases as

$$(a)\ \Delta l_q = \frac{F_{\mathrm{cab},q}}{r_c \cdot c_q^{'}} + b_q \Delta l_{K,q}, \tag{6.14}$$

$$(b)\ \Delta l_q = \frac{F_{\mathrm{cab},q}}{c_q} + \frac{b_q^{'}}{r_c} \Delta l_{K,q}, \tag{6.15}$$

$$(c)\ \Delta l_q = \frac{F_{\mathrm{cab},q}}{r_c \cdot c_q^{'}} + \frac{b_q^{'}}{r_c} \Delta l_{K,q}, \tag{6.16}$$

$$(d)\ \Delta l_q = \frac{F_{\mathrm{cab},q} \cdot r_c}{c_q^{''}} + \frac{b_q^{'}}{r_c} \Delta l_{K,q}, \tag{6.17}$$

$$(e)\ \Delta l_q = \frac{F_{\mathrm{cab},q}}{r_c \cdot c_q^{'}} + b_q^{''} \cdot r_c \Delta l_{K,q}. \tag{6.18}$$

The mean and standard deviation of e_{avg} and $e_{\max}$ are displayed in Fig. 6.6 and Fig. 6.7 for all five ideas and approach 1 and the trajectories given in Tab. 6.2. The results show that in general for smaller radii the errors increase. Case (c) shows worst results for small radii and (d) for larger radii. The case with the smallest overall errors is given by case (a), showing that scaling of parameter c_q by considering the radius of the circle r_c seems reasonable for larger circles. Adjusting the parameter b_q by the radius purely results in stronger deviation. The standard deviation is very small for both errors.

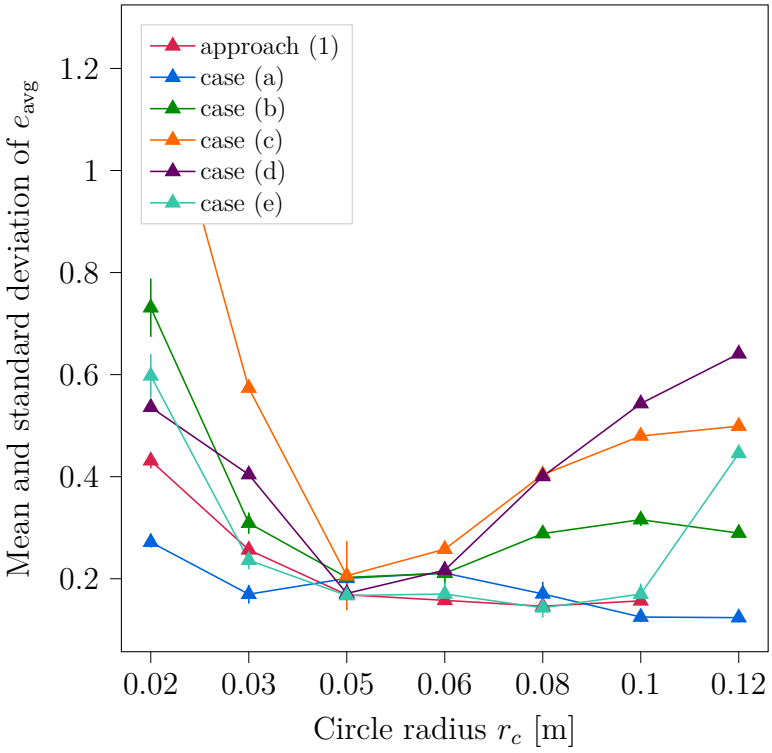


Figure 6.6: Mean and standard deviation of e_{avg} for circles of different size.

Approach 3: Function Fitting

The last approach presented in this work will fit a function through the measured parameters for a handful of chosen circles. As seen for the previous approach scaling with the radius does not seem to improve the position tracking much. It could be possible to potentially achieve a radius-scaling function, where larger and smaller radii are distinguished, but a promising result is uncertain and therefore will not be examined further. Instead, the idea is introduced to tune individual parameters for a handful of chosen circles with different radii. Then a polynomial of optional degree can be derived, which will fit the collected values for each parameter of a different circle onto a curve. This results in a function of the current distance between the robot's desired tip and center

$$c_q = f_q(\|\tilde{\boldsymbol{z}}_t\|) \tag{6.19}$$
$$b_q = g_q(\|\tilde{\boldsymbol{z}}_t\|). \tag{6.20}$$

If the 6 functions f_q and g_q for $q = 1 \dots 3$ are tuned adequate, the robot will be able to reach any desired position, which enables tracking of arbitrary trajectories.

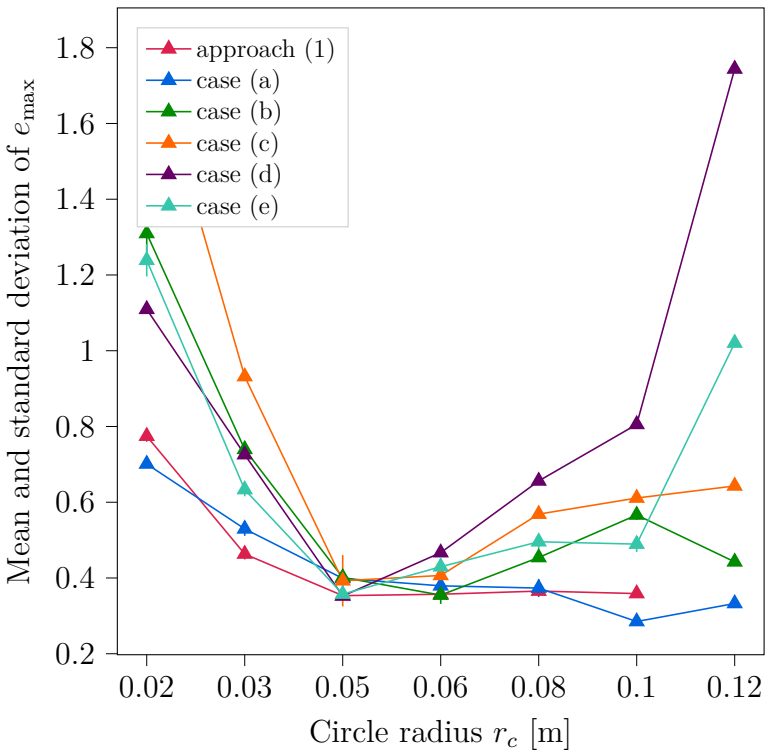


Figure 6.7: Mean and standard deviation of $e_{\max}$ for circles of different size.

However, as already seen in Fig. 6.4, Fig. 6.6 and Fig. 6.7 the errors for trajectories with smaller radii are larger than for motion with a greater distance to the origin. Therefore the two functions are determined for larger radii only and this approach will be used for examining the influence of dynamic properties on triangular trajectories in Sect. 6.2, because for any non-circular trajectory or circle, whose center point is not located at the origin, $\|\tilde{\boldsymbol{z}}\| \neq r_c$ and therefore the scaling might be insufficient. Additionally, with the function representation specific parameters for the chosen circles can be tuned and therefore potentially increase the performance.

For deriving all 6 functions f_q and g_q with $q = 1 \ldots 3$ circular trajectories are considered of $r_c \in \{8\,\text{cm}, 10\,\text{cm}, 12\,\text{cm}\}$. For these radii the robot manages to stay consistently on the desired path for well tuned parameters. The according tuned parameters are listed in Tab. 6.5.

A polynom of second order is chosen to approximate the parameter functions f_q and g_q for $q = 1 \ldots 3$. The resulting polynomials f_q are displayed in Fig. 6.8 and the polynomials g_q are visualized in Fig. 6.9.

Table 6.5: Tuned parameters for circular paths of different sizes.

Parameter	$r_c = 8\,\text{cm}$	$r_c = 10\,\text{cm}$	$r_c = 12\,\text{cm}$
c_1 in $\frac{\text{N}}{\text{m}}$	1350	1400	1450
c_2 in $\frac{\text{N}}{\text{m}}$	1100	1050	1000
c_3 in $\frac{\text{N}}{\text{m}}$	1050	1300	1600
b_1	1.05	1.14	1.27
b_2	1.15	1.13	1.15
b_3	1.2	1.25	1.3

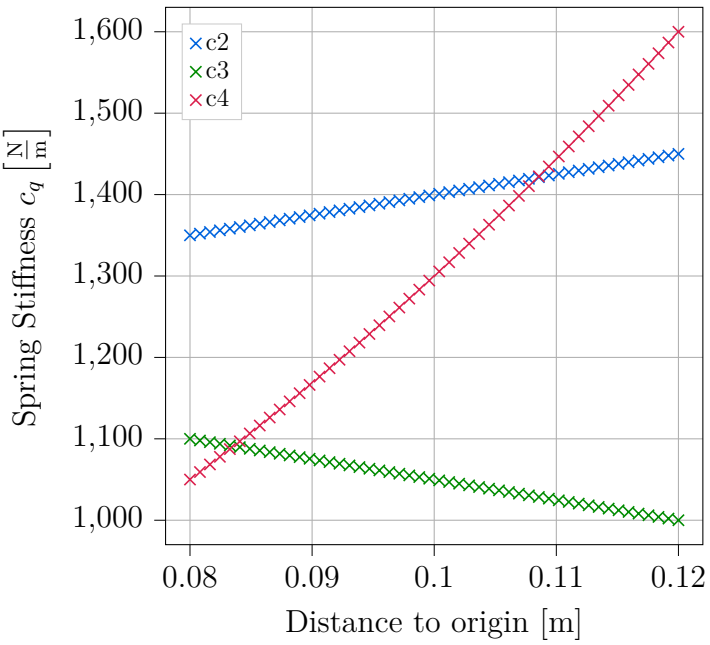


Figure 6.8: Parameter functions for c_q with $q = 1 \ldots 3$.

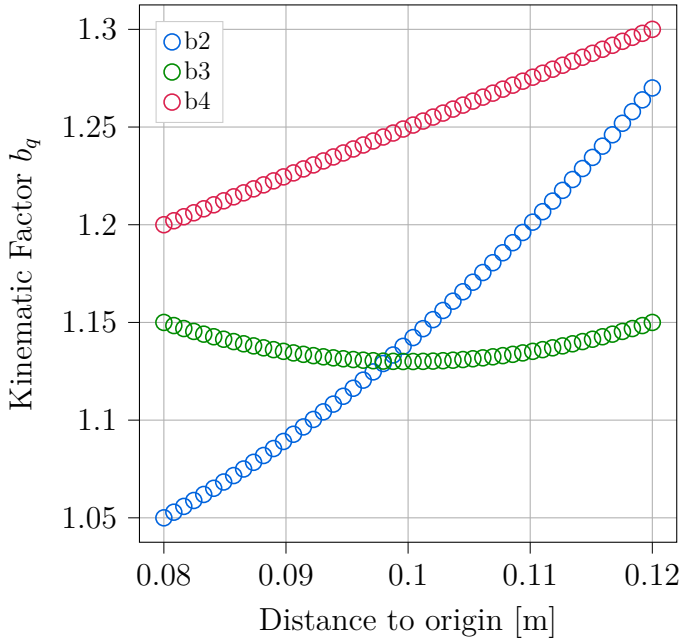


Figure 6.9: Parameter functions for b_q with $q = 1 \dots 3$.

6.2 Trajectory Following

The second half of this chapter is dedicated to examination of the influence of user-chosen parameters on the robots dynamic behavior. Two types of trajectories are considered for this analysis, namely circular and triangular paths. The third approach to determine the parameters is applied to triangular trajectories in Sect. 6.2.1, while for simple circlular paths parameters are tuned for the corresponding examined radii r_c. Three parameters and their influence on trajectory following are studied, namely the step size Δt of the implicit Euler algorithm, see Eq. (4.27), for solving the inverse model Eq. (4.25), the path velocity of the desired trajectory $\widetilde{v}$ and choice of number of segments N for modeling the robot.

The studied settings for all three parameters Δt, v_{lim} and N are given in Tab. 6.6. Note, that the step size is equal to the transmission frequency for sending inputs to the motors.

Table 6.6: Values for user-chosen variables.

Δt in s	0.1	0.02	0.01
v_{lim} in $\frac{m}{s}$	0.05	0.075	0.1
N	1	3	6

Circular Trajectory

The first parameter, which can effect the dynamic behavior of the robot, is the step size Δt for solving the governing DAE equations Eq. (4.25) and therefore rate of inputs for the actuation. As already explained in Sect. 5.2, in some cases the solving algorithm struggles to find a solution. This usually happens for specific combination of desired velocities v_{lim} and step sizes Δt. But by slight adjustment of the initial deflection direction the solver was able to find solutions and integrate Eq. (4.27).

The settings for the circular trajectories, used to study dynamic behavior, are listed in Tab. 6.7. The smallest circle is of size $r_c = 4\,\text{cm}$ because for smaller circles the robot does not manage to consistently follow the circle and disturbances would have greater impact, as explained in Sect. 6.1.2. The largest radius of $r_c = 12\,\text{cm}$ was chosen because it marks the task space limit. Lastly, the circle of size $r_c = 8\,\text{cm}$ serves as an intermediate.

Table 6.7: Circular Trajectories for trajectory following.

Shape	Size	max. velocity	Duration (initial)	Number of laps
-	r_c [cm]	$v_{\max}$ $[\frac{\text{m}}{\text{s}}]$	t_{init} [s]	n_{lap}
Circle	4	0.05	3	4
Circle	4	0.075	3	4
Circle	4	0.1	3	4
Circle	8	0.05	3	4
Circle	8	0.075	3	4
Circle	8	0.1	3	4
Circle	12	0.05	3	4
Circle	12	0.075	3	4
Circle	12	0.1	3	4

Note, that the maximum velocities $v_{\max}$ of the constant velocity profile are equal to the desired velocities listed in Tab. 6.6. The maximum acceleration $a_{\max}$ and maximum jerk $j_{\max}$ are chosen freely with reasonable values. But as both values are not examined in this work, their value has to be chosen to generate a desired trajectory with a constant velocity profile, as explained in Sect. 5.3.3. Their values are chosen to be $a_{\max} = 0.1\,\frac{\text{m}}{\text{s}^2}$ and $j_{\max} = 0.1\,\frac{\text{m}}{\text{s}^3}$. Note, that for the dynamic experiments a series of 2 experiments was taken for each configuration.

The results show that for faster path velocities v_{lim} both errors increase, see Fig. 6.10-Fig. 6.12. For $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$ the mean of the averaged, normed tip error shows small deviation for different step size Δt, depicted on the left side in Fig. 6.10. In this case the robot follows the circular trajectory rather slow and dynamic effects are small. Similarly, the mean of the maximum normed tip error deviates on a small scale for different Δt, seen on the right in Fig. 6.10. There are only minor changes noticeable for different Δt in terms of tip accuracy. The larger standard deviation of e_{max} for $\Delta t = 100\,\text{s}$ is traced back on the small number of experiments for each series.

In comparison, $v_{\text{lim}} = 0.075\,\frac{\text{m}}{\text{s}}$ shows larger error values for each radius, see Fig. 6.11. Especially for a radius of $r_c = 4\,\text{cm}$ the errors are much higher compared to the previous case. Again, the step size Δt has only minor effect on the average tip error e_{avg}. Contrarily, the maximum tip error e_{max} is more impacted by the step size Δt. However, the deviation is still very small for different step sizes Δt.

Last, for $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$ the errors increase again, see Fig. 6.12. The average error e_{avg} does increase only little compared to the maximum error e_{max}. The step size doesn't seem to have large effect. Only for a very large radius r_c the average

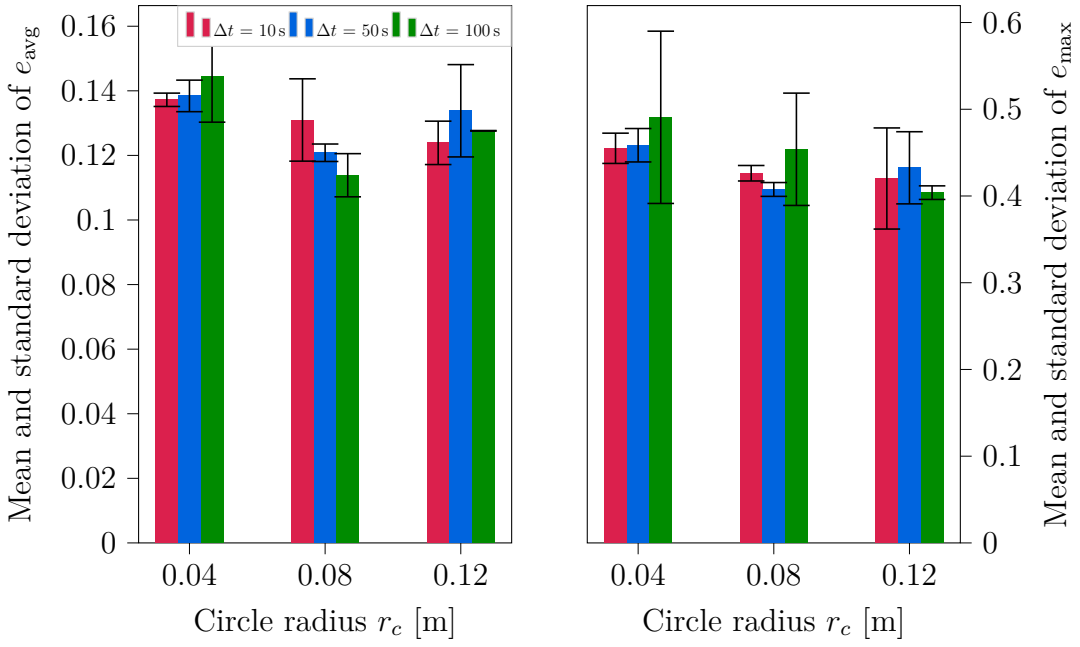


Figure 6.10: Mean and standard deviation of e_{avg} and e_{max} for $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$ and different step sizes Δt.

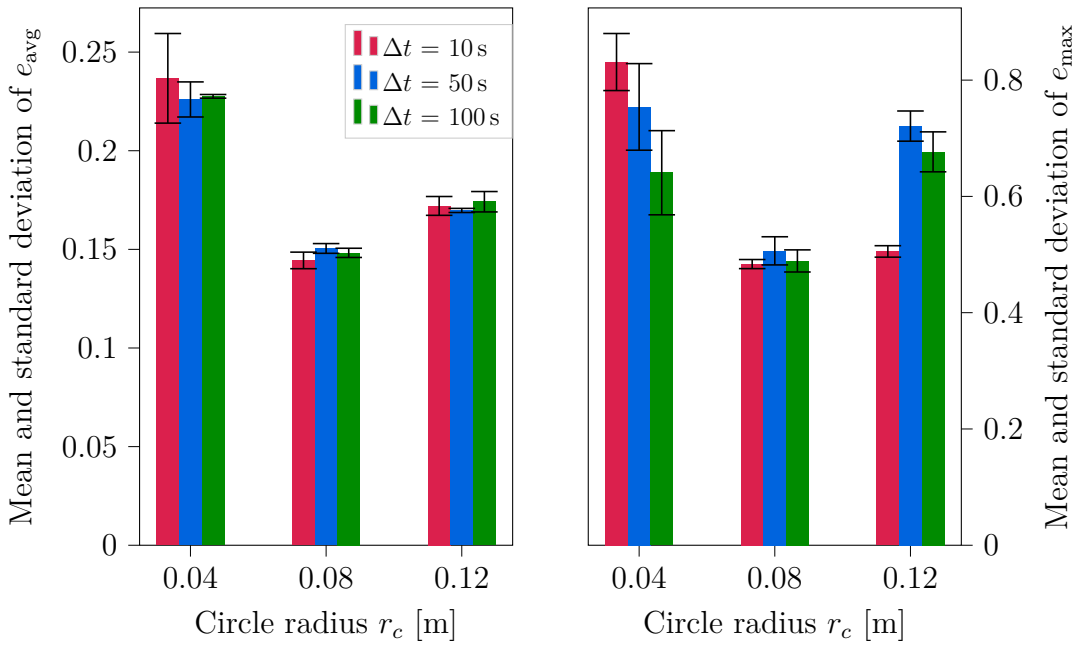


Figure 6.11: Mean and standard deviation of e_{avg} and e_{max} for $v_{\text{lim}} = 0.075\,\frac{\text{m}}{\text{s}}$ and different step sizes Δt.

tip error e_{avg} is higher for $\Delta t = 100\,\text{s}$. However, again this can be traced back on the small amount of experiments.

The error plots show a commonality. In most cases, the errors for a trajectory with $r_c = 8\,\text{cm}$ have the lowest values. The reason for this is that a trajectory with $r_c = 4\,\text{cm}$ is closer to the origin, where the disturbance variables have a greater influence, as explained in Sect. 6.1.2. For a trajectory with $r_c = 12\,\text{cm}$ the robot reaches its work-space limit.

The recorded trajectory for $r_c = 4\,\text{cm}$ is shown in Fig. 6.13. After the acceleration phase the robot moves with a constant speed of $v_{\text{lim}} = 0.075\,\frac{\text{m}}{\text{s}}$. The step size is $\Delta t = 0.02\,\text{s}$ and the robot rotates $n_{lap} = 4$ times around its center axis. Only a limited amount of measurements is available. This is due to the rather fast speed of $v_{\text{lim}} = 0.075\,\frac{\text{m}}{\text{s}}$. The camera used for tracking struggles to detect the AprilTags, which leads to the large gaps in between the measurements. At these positions the AprilTags cube is rotated by unfavorable angles, where together with the higher speed the camera struggles to detect the blurry AprilTags. The initial deflection shows the opposite effect, when the cube moves out of the straight configuration. In this case, the cube was rotating only around one axis and therefore tracking more successful. However, the measurements still show that the robot is capable of following the circular

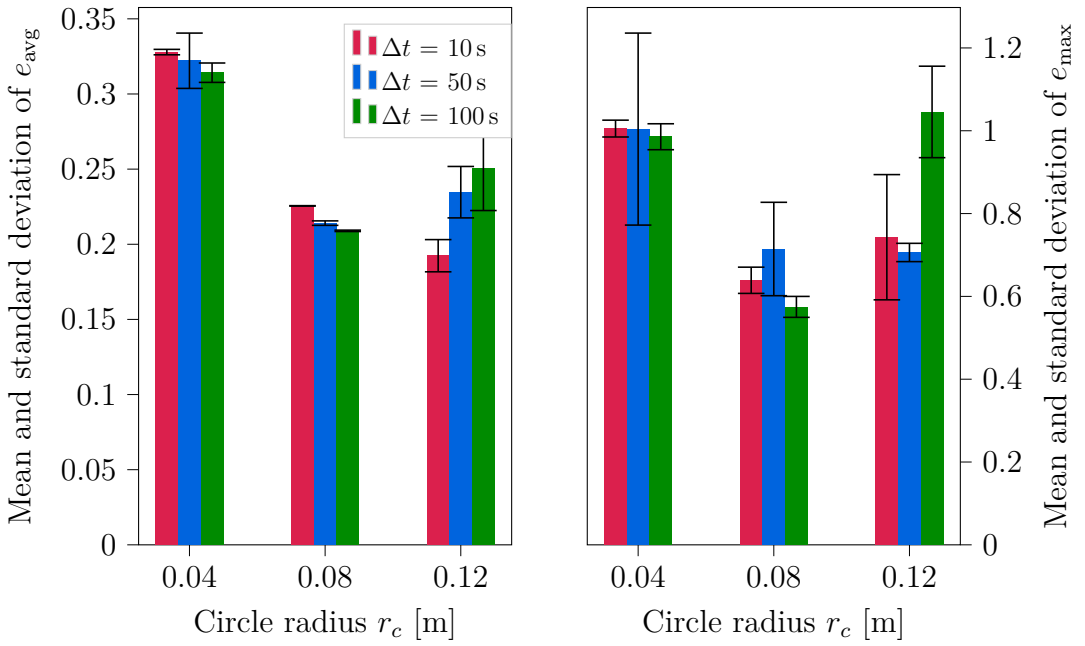


Figure 6.12: Mean and standard deviation of e_{avg} and e_{max} for $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$ and different step sizes Δt.

trajectory in general.

In Fig. 6.14 the case with $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$ and $\Delta t = 0.01\,\text{s}$ for a trajectory of size $r_c = 8\,\text{cm}$ is visualized. Again, the robot takes $n_{lap} = 4$ laps and follows a constant velocity profile after initial acceleration. At an even higher speed the camera struggles further to detect the cube. The gaps are larger compared to Fig. 6.13, but the measurement accumulations are clearly more dense because of the higher input transmission frequency (smaller $\Delta t = 0.01\,\text{s}$). Again, even for a considerably high speed $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$ the robot still manages to follow the desired trajectory approximately.

At last, Fig. 6.15 shows the largest trajectory of $r_c = 12\,\text{cm}$. After initial acceleration, the desired constant speed $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$ is slower compared to the previous examples. The step size is $\Delta t = 0.1\,\text{s}$ and again the robot takes $n_{lap} = 4$ laps. The plot shows how the robot accurately tracks the desired circle. Especially when motor 1 and 3 actuate mainly the robot manages to follow the desired path very accurately in all 4 laps. Only in the lower left corner, where motor 2 is the main actuator, the robot struggles to stay on track. In one of the four lap, the robot cuts short strongly, compared to the other laps. This is due to the limiting task space of the robot. While the robot would be able to reach even wider distances in theory, at a size of $r_c = 12\,\text{cm}$ motor 2 (lower left)

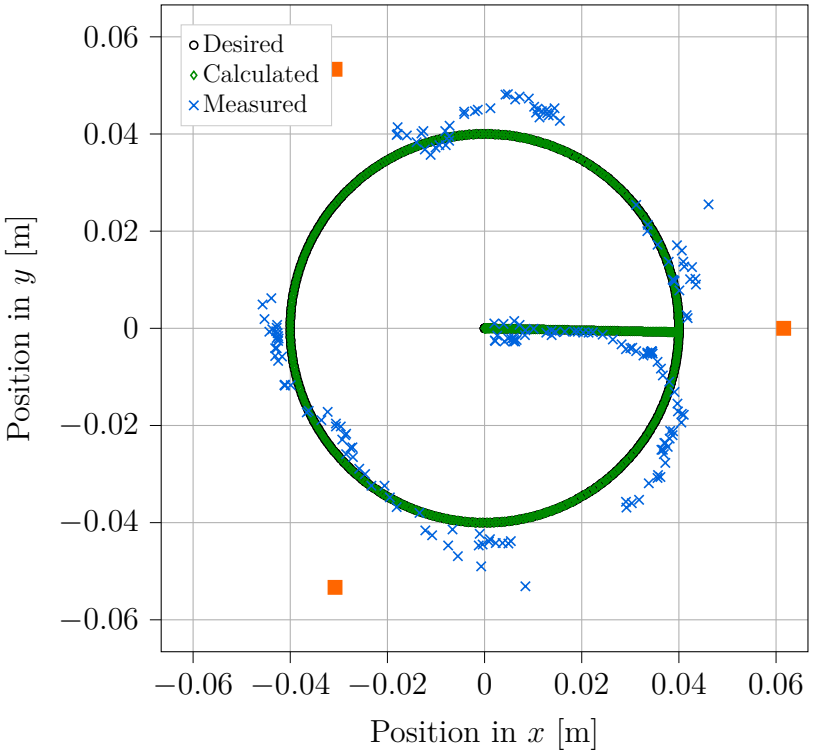


Figure 6.13: Trajectory for $r_c = 4\,\text{cm}$ with $\Delta t = 0.02\,\text{s}$ and $\widetilde{v} = 0.075\,\frac{\text{m}}{\text{s}}$, measured (blue), calculated (green) and desired (black).

struggles to apply the required strain in order to bend the robot. The reason, why only the motor 2 reaches its limit, is due to initial strain on the cables for the state of rest. During mounting it is necessary to ensure a small load is induced on the robot by each cable. If the cables are lose, when the robot rests, elongation will have a larger effect at the start and initial load of smaller values gets lost. In that case, the robot performs even more worse on trajectories of smaller size

The limiting case is noticeable even stronger when looking at the tracked x-positions in Fig. 6.16 and y-positions in Fig. 6.17 over time. The largest short cut happens around $t = 52\,\text{s}$, as seen in Fig. 6.15. Accordingly, at $t = 52\,\text{s}$ the robot falls short in negative y-direction, as can be seen in Fig. 6.17, where the tracked y position doesn't reach the maximum of the desired track.

Further the effects of the step size Δt on the velocity v_{lim} can be studied. The considered step sizes Δt are given in Tab. 6.6. A circle of size $r_c = 8\,\text{cm}$ is chosen, as for most cases it shows the smallest errors for variation of the step size Δt and the velocity v_{lim}. The positional error is defined as

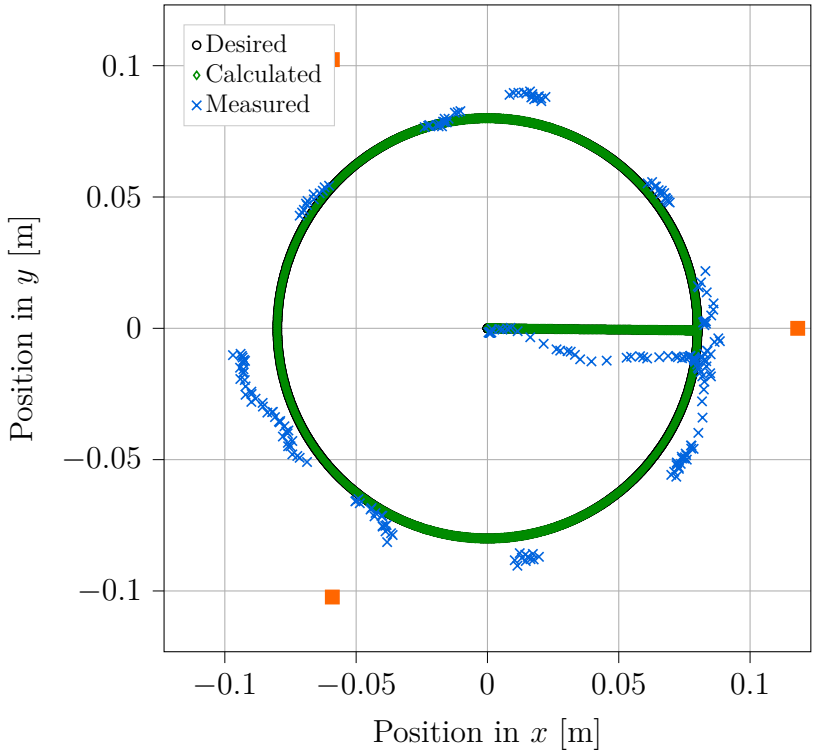


Figure 6.14: Trajectory for $r_c = 8\,\text{cm}$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$, measured (blue), calculated (green) and desired (black).

$$e_{\text{pos}} = \|\tilde{\boldsymbol{z}}_t - \boldsymbol{z}^{\text{meas}}\|. \tag{6.21}$$

The error is visualized in Fig. 6.18, once containing the aforementioned time delay, depicted in red, and once more delay-clean, colored in purple. The time delay is removed by shifting the measured data in order to match with the desired path approximately. As seen, the time delay has a significant effect on the positional error.

The velocity, obtained by time integration with measured and calculated time points, is depicted in Fig. 6.22 for $\Delta t = 0.01\,\text{s}$, in Fig. 6.21 for $\Delta t = 0.05\,\text{s}$ and in for $\Delta t = 0.1\,\text{s}$. The robot moves with the desired velocity at the majority of time, but larger peaks appear in the tracked velocity. These can be traced back to jumps in the measured positions and numerical noise. Already minor offsets will lead to jumps in the velocity function. For example, the largest peak in Fig. 6.19 is computed with the measured time difference $\Delta t_{\text{meas}} = 16.7207\,\text{s} - 16.6372\,\text{s}$

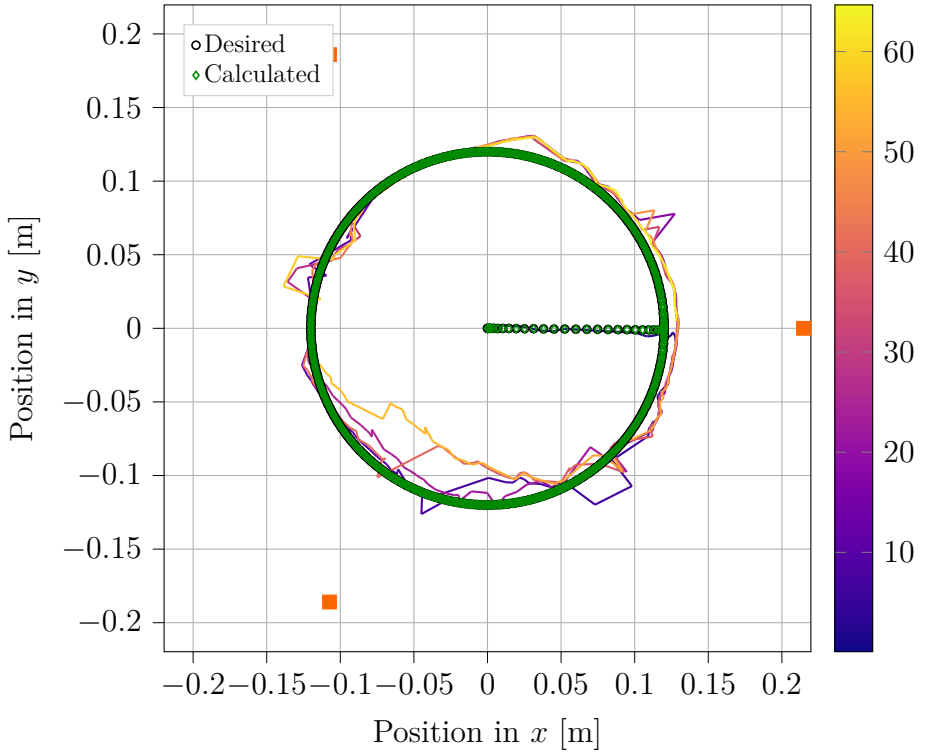


Figure 6.15: Trajectory for $r_c = 12\,\text{cm}$ with $\Delta t = 0.1\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (blue), calculated (green) and desired (black).

$$v_{\text{max}} = \frac{\sqrt{(0.0036\,\text{m} - 0.0139\,\text{m})^2 + (-0.0553\,\text{m} + 0.0799\,\text{m})^2}}{\Delta t_{\text{meas}}} = 0.319\,\frac{\text{m}}{\text{s}}. \quad (6.22)$$

Vibrations and measurement noise of the robot lead to jumps of the measured x and y-positions of the robot's tip. The oscillating behavior is natural for the silicone material, the robot is made of. Measurement noise occur due to AprilTags not being detected or an inaccurate, and sometimes even incorrect, rotation is computed by the detection algorithm. Already rather smaller jumps of $1\,\text{cm} \sim 2\,\text{cm}$ lead to strong peaks in the velocity profile. Therefore, these peaks are filtered and not included in the visualization Fig. 6.22 - Fig. 6.19. But it can be seen that the step size Δt does not influence the velocity of the robot significantly.

In terms of the desired path velocity v_{lim}, Fig. 6.23 and Fig. 6.24 show trajectories with $v_{\text{lim}} = 0.075\,\frac{\text{m}}{\text{s}}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$. Because the camera detects less AprilTags for higher velocities, the step size is kept high at $\Delta t = 0.01\,\text{s}$. The robot does manage to keep the desired velocity more accurate with less

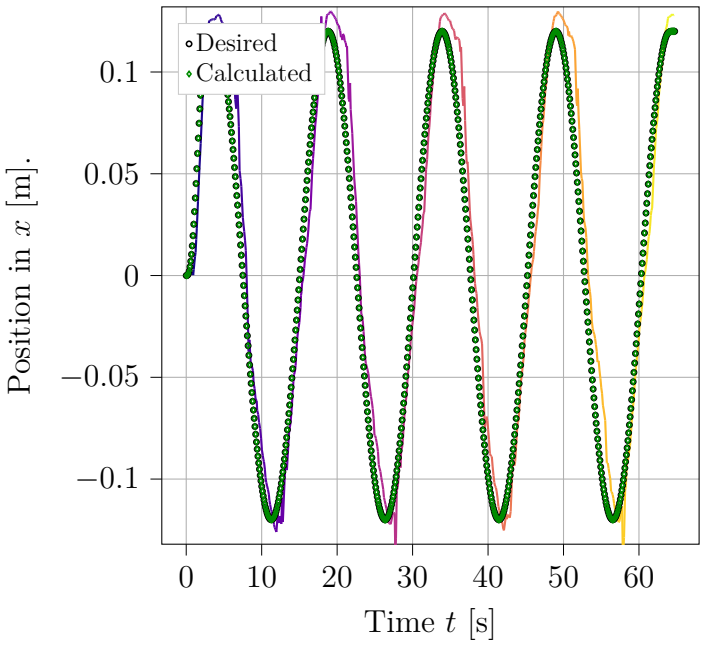


Figure 6.16: Position in x over time for $r_c = 12\,\text{cm}$ with $\Delta t = 0.1\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

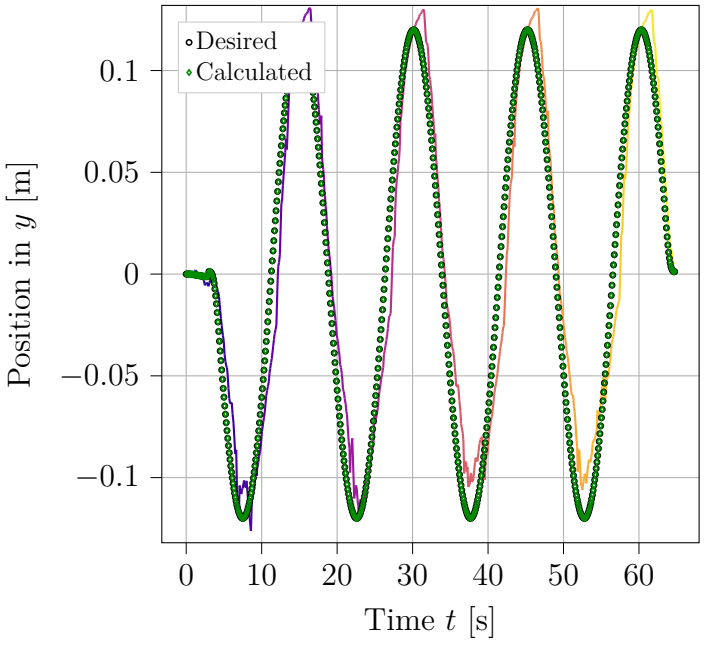


Figure 6.17: Position in y over time for $r_c = 12\,\text{cm}$ with $\Delta t = 0.1\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

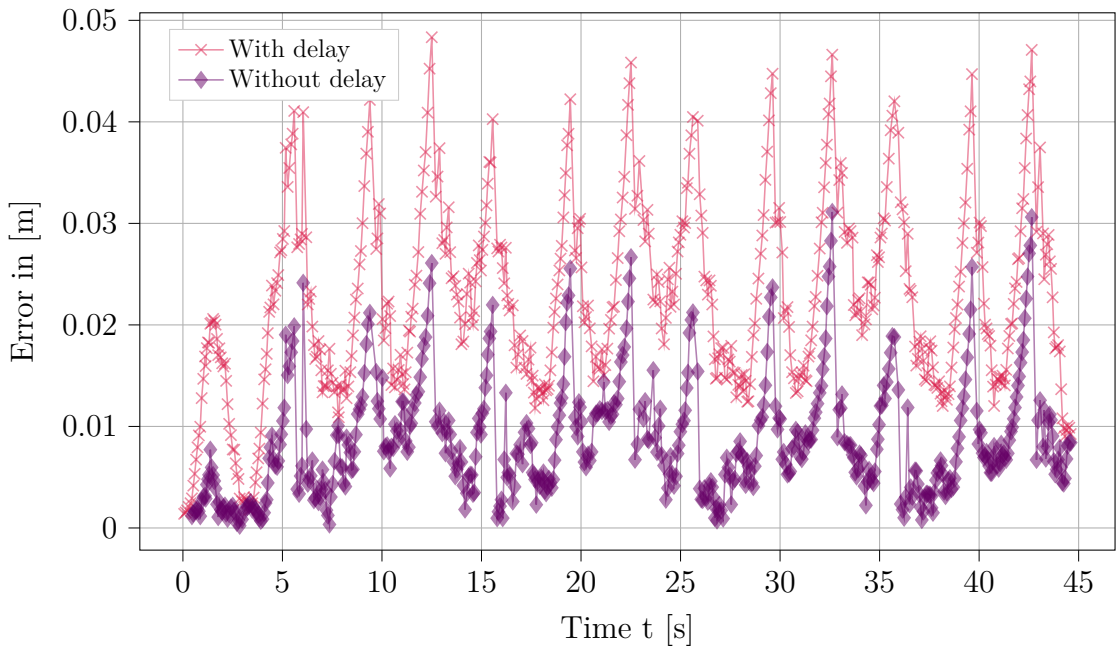


Figure 6.18: Positional error over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, with time delay (red), without (purple)

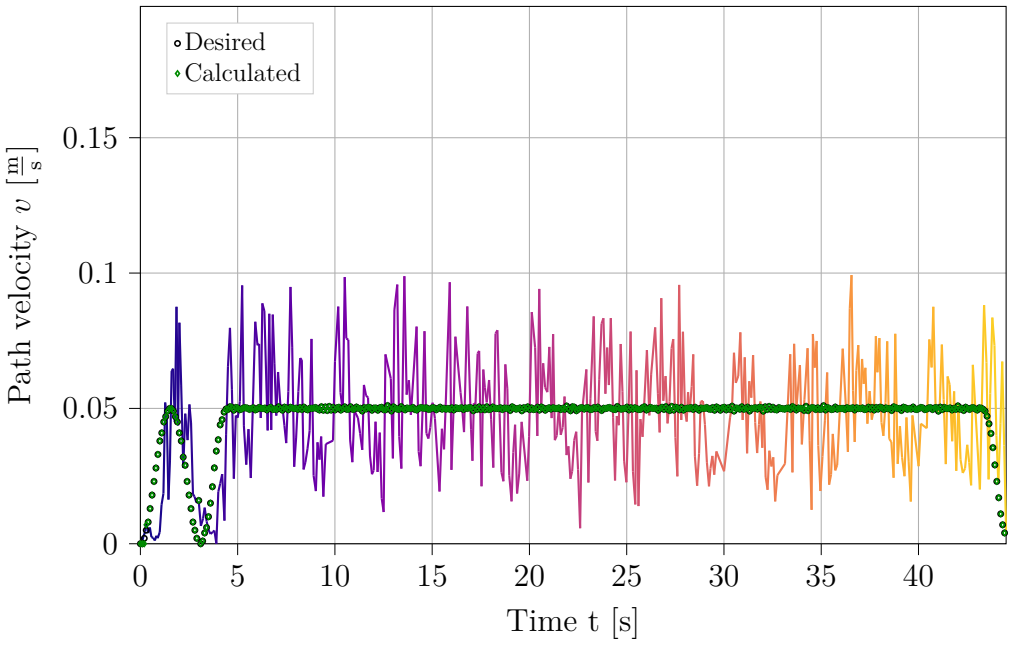


Figure 6.19: Velocity over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.1\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

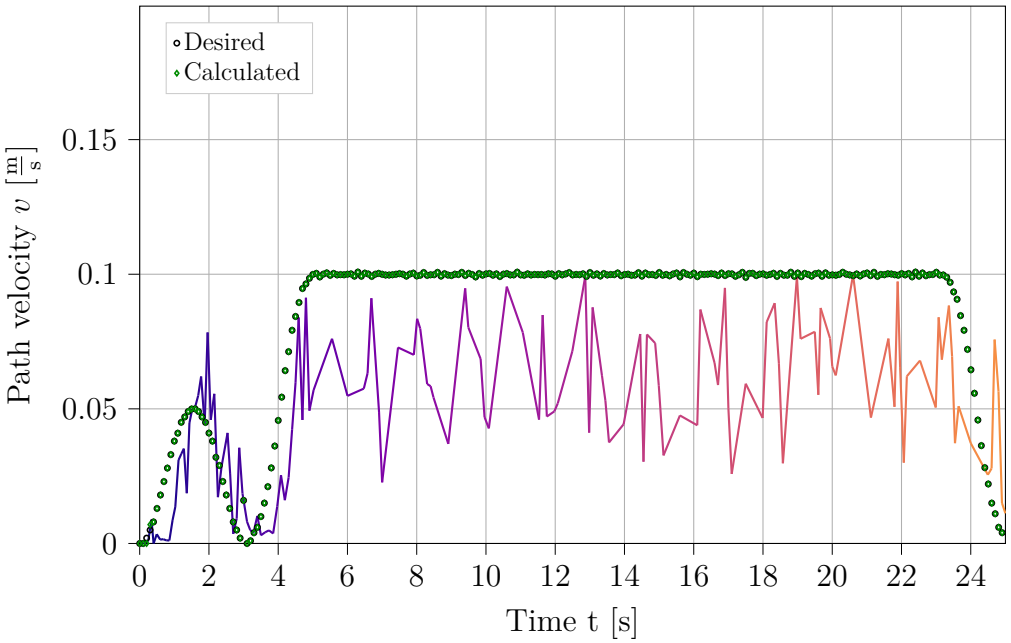


Figure 6.20: Velocity over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.1\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

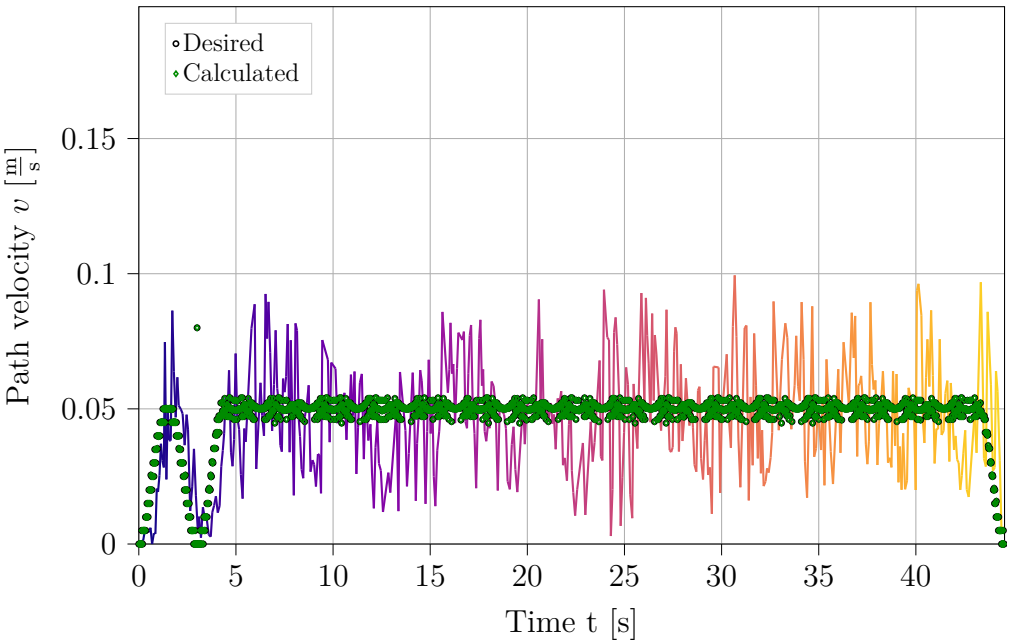


Figure 6.21: Velocity over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.05\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

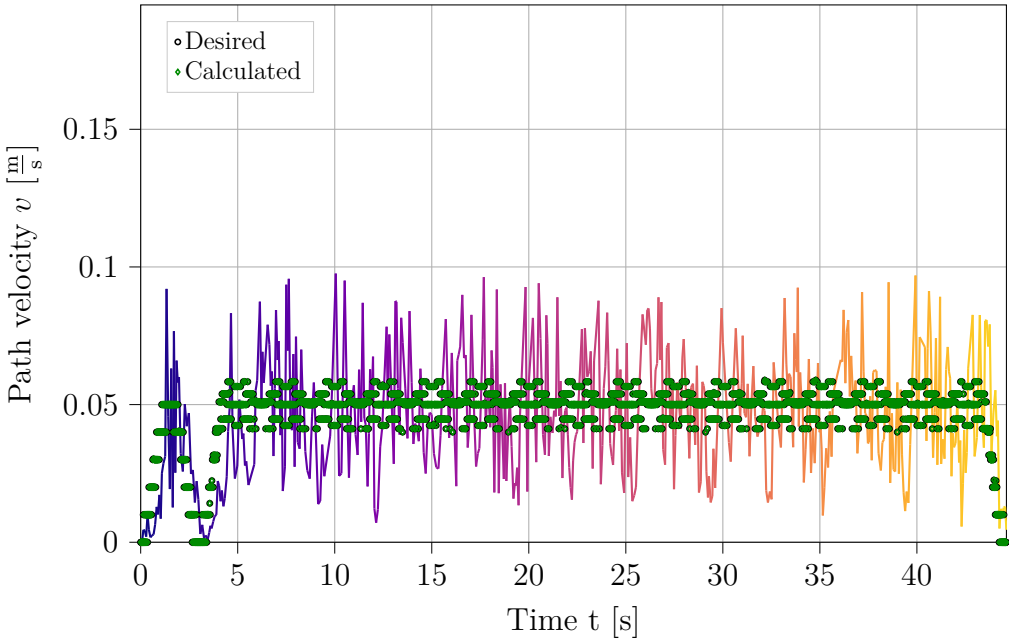


Figure 6.22: Velocity over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

stronger peaks, especially for $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$. However this is due to the overall higher velocity and inaccuracies of the position measurement do not stand out as significantly as for lower velocities.

Finally, the analysis is completed by looking at the number of segments N. The considered values for N are listed in Tab. 6.6. A trajectory with constant velocity profile is considered with $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$ to examine how the number of segments affect the results for fast motion, together with a small $\Delta t = 0.01\,\text{s}$. The tracked velocities are depicted in Fig. 6.25, Fig. 6.26 and Fig. 6.27 for $N = 1$, $N = 3$ and $N = 6$, respectively. It is clearly observable how the tracking of the desired trajectory becomes worse for a smaller number of segments to model the softrobot. Therefore, for path tracking with high velocities an appropriate number of segments is favorable to obtain accurate trajectory tracking.

At last, Fig. 6.28 shows the motor angles over time. The green graph considers the motor angle α_q at time t and the blue graph gives information about the influence of the change in length of the virtual $\Delta l_{F,q}$ in the motor α_q. Here the oscillation for following a circle and actuation delay between the motors are clearly visible.

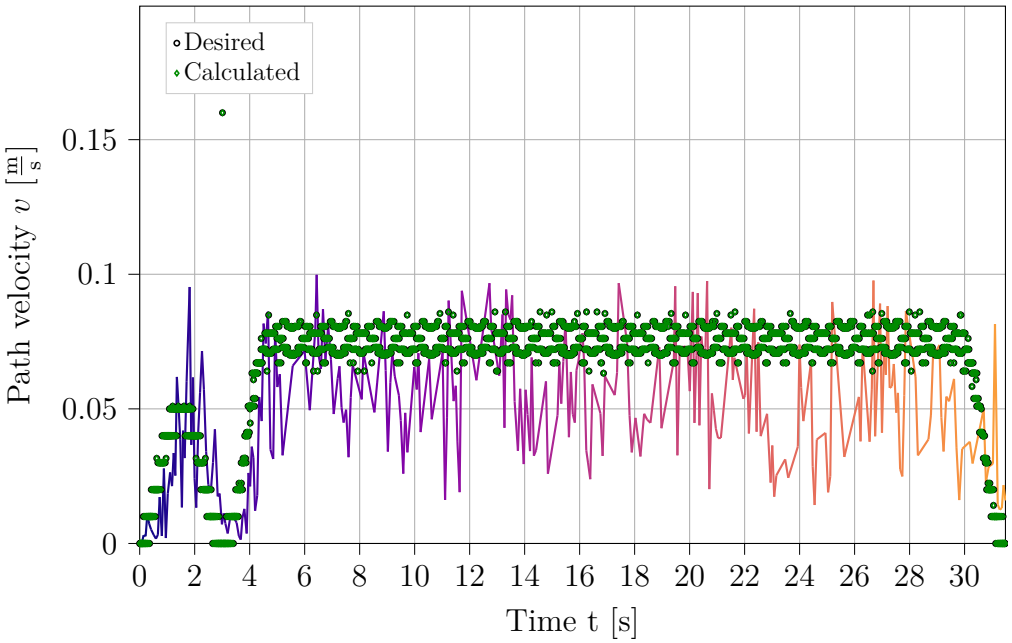


Figure 6.23: Velocity over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.075\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

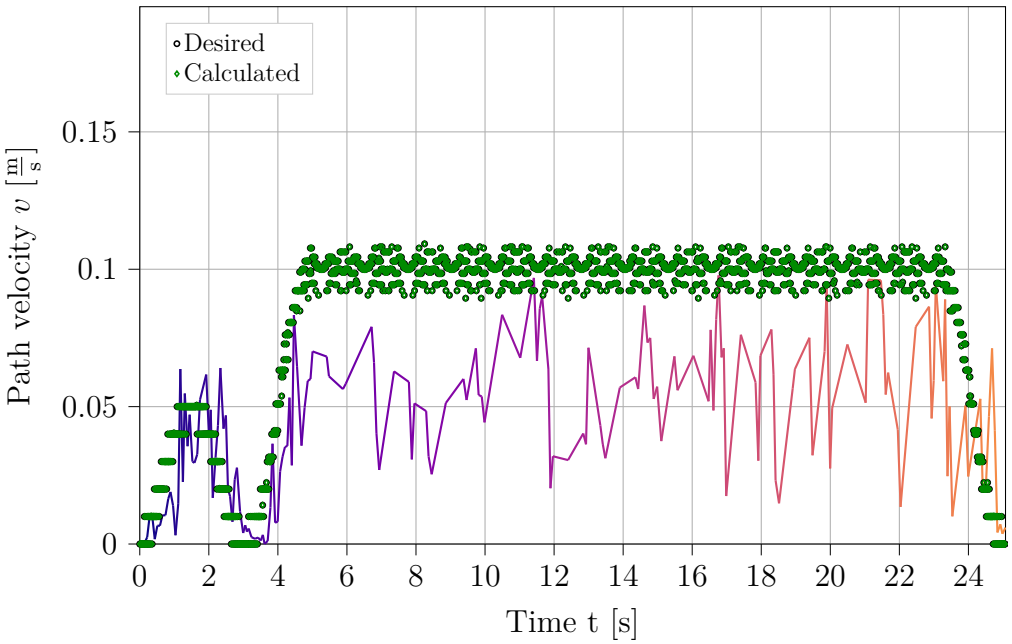


Figure 6.24: Velocity over time for $r_c = 8\,\text{cm}$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

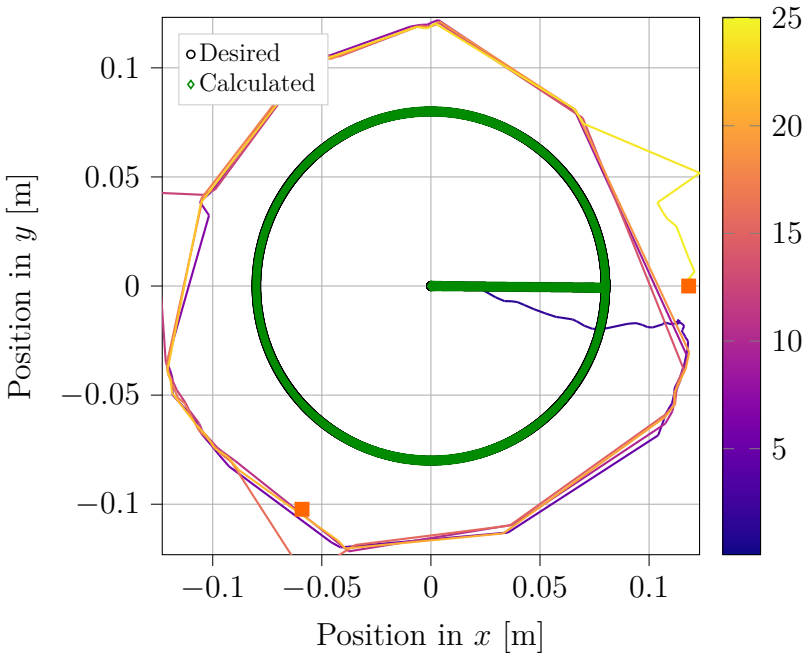


Figure 6.25: Circular trajectory of $r_c = 8\,\text{cm}$ for $N = 1$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

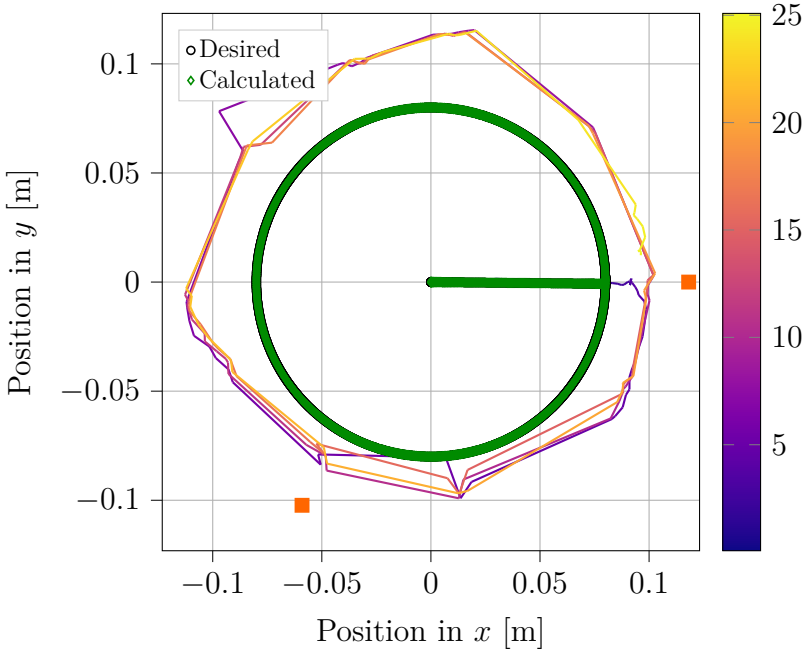


Figure 6.26: Circular trajectory of $r_c = 8\,\text{cm}$ for $N = 3$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

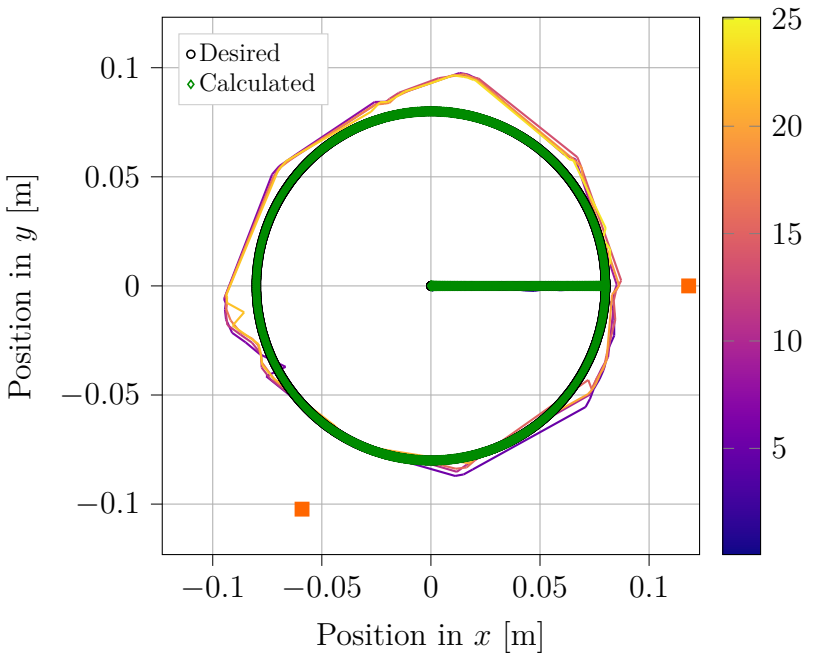


Figure 6.27: Circular trajectory of $r_c = 8\,\text{cm}$ for $N = 6$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

6.2.1 Triangular Trajectory

The second trajectory studied in this work is a triangular path around the robot's middle axis, derived in Sect. 5.3.1. This trajectory is chosen because it contains straight parts in the task space as well as sharper edges. The effects of centrifugal forces do not help the robot in this case for tracking the shape of the path. Additionally, the influence of the motors positioning around the robot can be examined by placing the corners of the triangle path right at the motors or in between. Further, the corners can be designed to be very sharp to study the robots tracking of sharp turns, especially at higher speed.

For representation of the input parameters c_q and b_q the third approach is considered, where polynomials are derived to allow adjustable parameters. In that regard, each point on the triangular trajectory should preferably have a minimum distance of 8 cm to the robot's center and not lay further than 12 cm away from the center. For a triangle with rounder corners this can be guaranteed for almost each point. However, in case of very sharp corners this requirement can not be met.

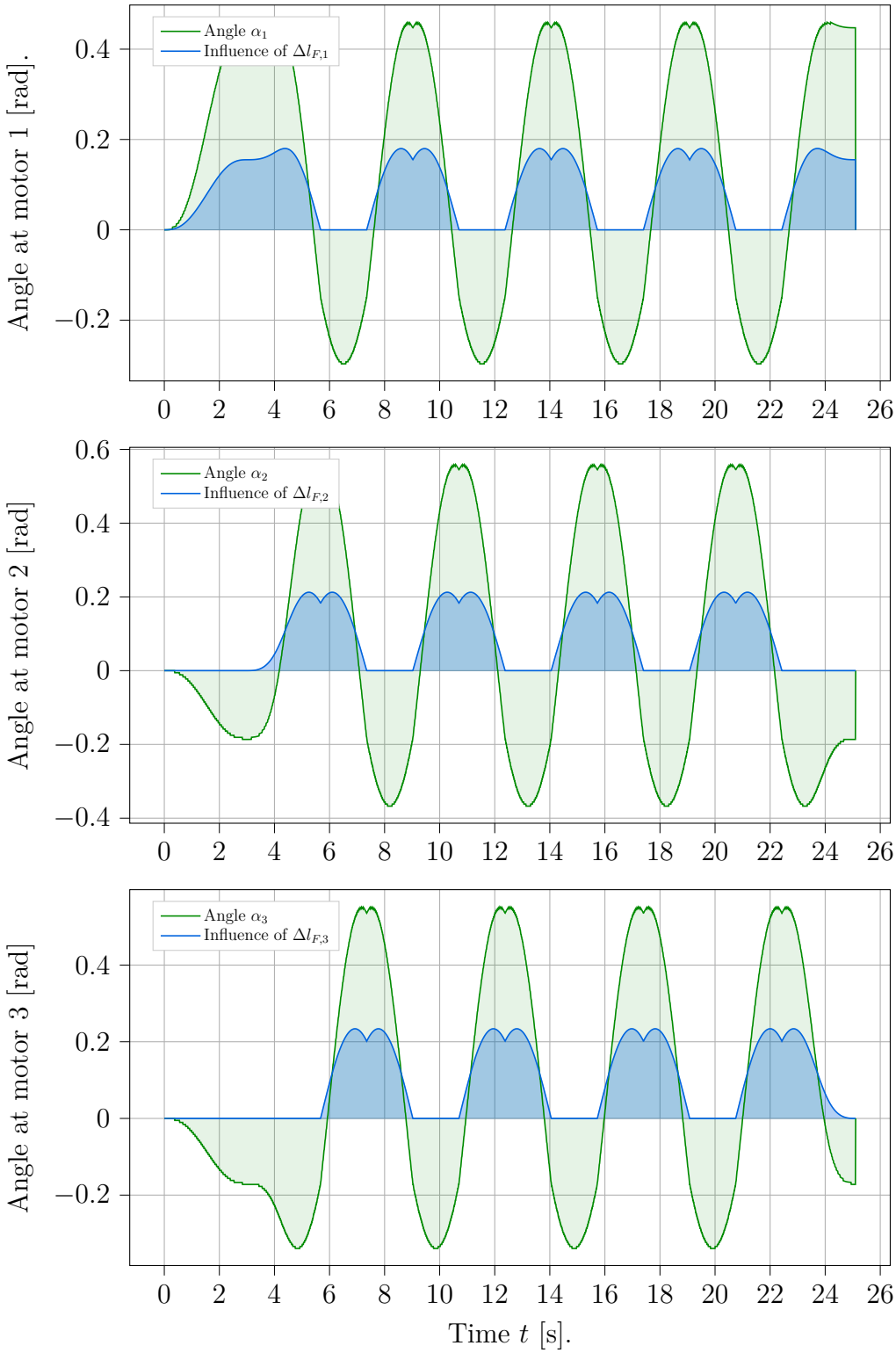


Figure 6.28: Motor angles α_q for $q = 1 \ldots 3$ over time t for a circle of size $r_c = 8\,\text{cm}$ with $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$.

Again, the step size Δt and velocity v_{lim} are examined with $\Delta t \in \{0.01\,\text{s}, 0.1\,\text{s}\}$ and $v_{\text{lim}} \in \{0.05\,\frac{\text{m}}{\text{s}}, 0.1\,\frac{\text{m}}{\text{s}}\}$. The chosen number of segments for modeling the robot is $N = 6$, as a smaller amount showed significantly poorer results. Three different trajectories are considered, a first one with rounded corners, where almost all points have a distance of at least 8 cm and at most 12 cm to the origin. Only the corners exceed this range slightly. The second one does get closer to the robots origin but possesses sharp edges. Lastly, the sharp triangle is rotated by 60°, moving the triangle's corners between two motors.

First, the triangle with round corners is considered. The tracked trajectory is visualized in Fig. 6.29 and Fig. 6.30 for $\Delta t = 0.1\,\text{s}$ with $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$ and $\Delta t = 0.01\,\text{s}$ and $v_{\text{lim}} = 0.1\,\frac{\text{m}}{\text{s}}$, respectively. The results show that the softrobot is capable of following the triangular trajectory accurately. Especially between the motors 1 and 3 the measurements show that the tip is very close to the desired positions.

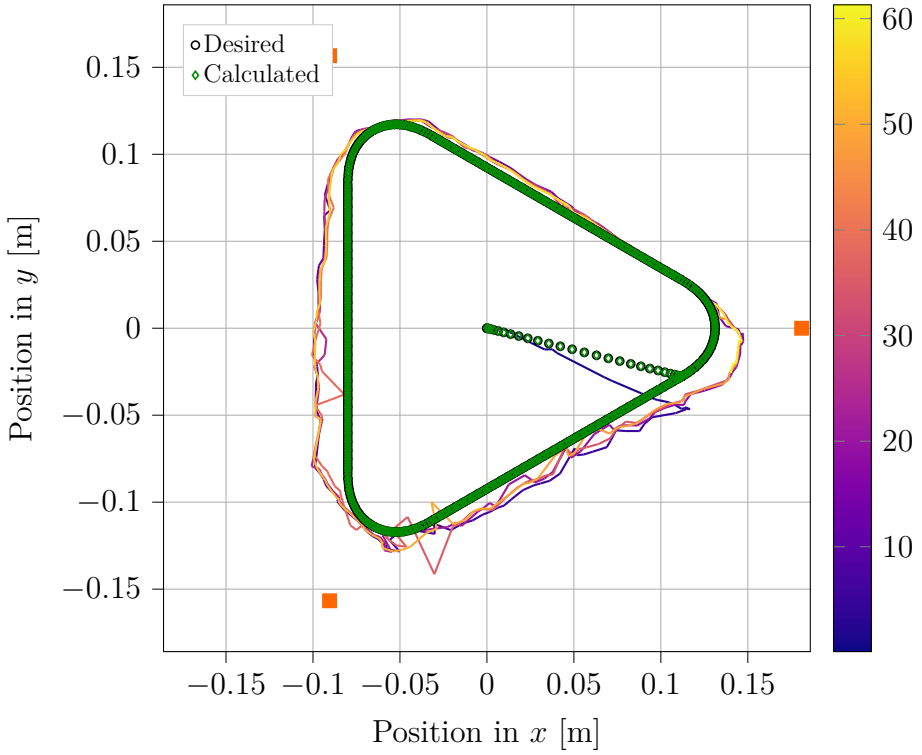


Figure 6.29: Triangular trajectory with round corners for $\Delta t = 0.1\,\text{s}$ and $v_{\text{lim}} = 0.05\,\frac{\text{m}}{\text{s}}$, measured (color gradient), calculated (green) and desired (black).

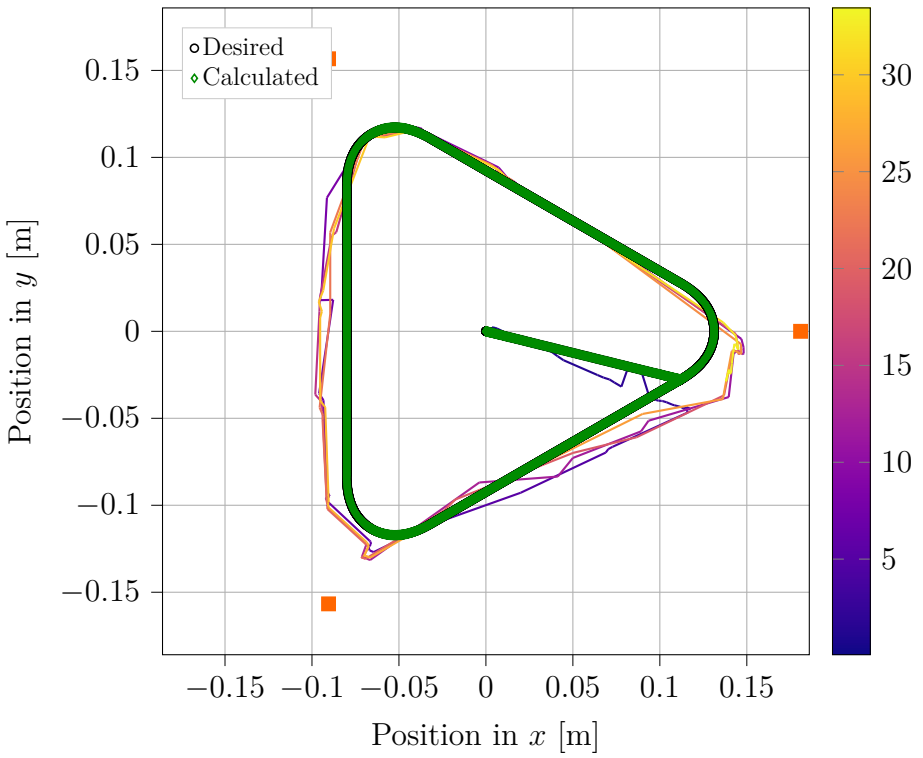


Figure 6.30: Triangular trajectory with round corners for $\Delta t = 0.01\,\mathrm{s}$ and $v_{\mathrm{lim}} = 0.1\,\frac{\mathrm{m}}{\mathrm{s}}$, measured (color gradient), calculated (green) and desired (black).

While both the limiting desired velocity v_{lim} and step size Δt do not seem to have any significant impact on the tracking results, it is worth noting that no short cuts are visible at motor 2, contrarily to the circular trajectory. Even though the desired positions of the corner section lay closer to the motor than compared to the circular case, the robot manages to reach the desired positions. This is due to the different actuation input over time. During the circular case a larger constant load is induced by the motors 1 and 3 to force the robot to stay on the circular path. However, for the corners of the triangular path the main actuator is motor 2 and smaller loads are induced by the other two motors, compared to the circular path. In particular for a higher velocity the robot struggles less to perform at the corner close to motor 2, since the inertia supports the motor more intense compared to slower speed. It is noticeable that in the case of a smaller desired velocity v_{lim} the robot does not cut short either, but approximates the corner arc more flatly.

The same effect is noticeable in the opposite way at the corner next to motor 1, where the robot exceeds the desired trajectory. This gap occurs because the

robot is initially pulled towards motor 1 to get on track, where the tuning was very challenging and therefore less accurate values were found. This effect is also slightly noticeable during the circular trajectory in Fig. 6.14, but is more distinct for the triangular path due to the corner, where motor 1 is the main actuator.

In general, the robot manages to follow the straight lines precisely with only smaller offsets. The round corners are tracked sufficiently, only for the corner at motor 1 a larger gap appears. To investigate the characteristic behavior at the corners further, a triangle with more sharp corners is considered. Again, a step size of $\Delta t = 0.1\,\mathrm{s}$ together with a path velocity of $v_{\mathrm{lim}} = 0.05\,\frac{\mathrm{m}}{\mathrm{s}}$ and a step size of $\Delta t = 0.01\,\mathrm{s}$ in addition to a velocity of $v_{\mathrm{lim}} = 0.1\,\frac{\mathrm{m}}{\mathrm{s}}$ are shown in Fig. 6.31 and Fig. 6.32, respectively.

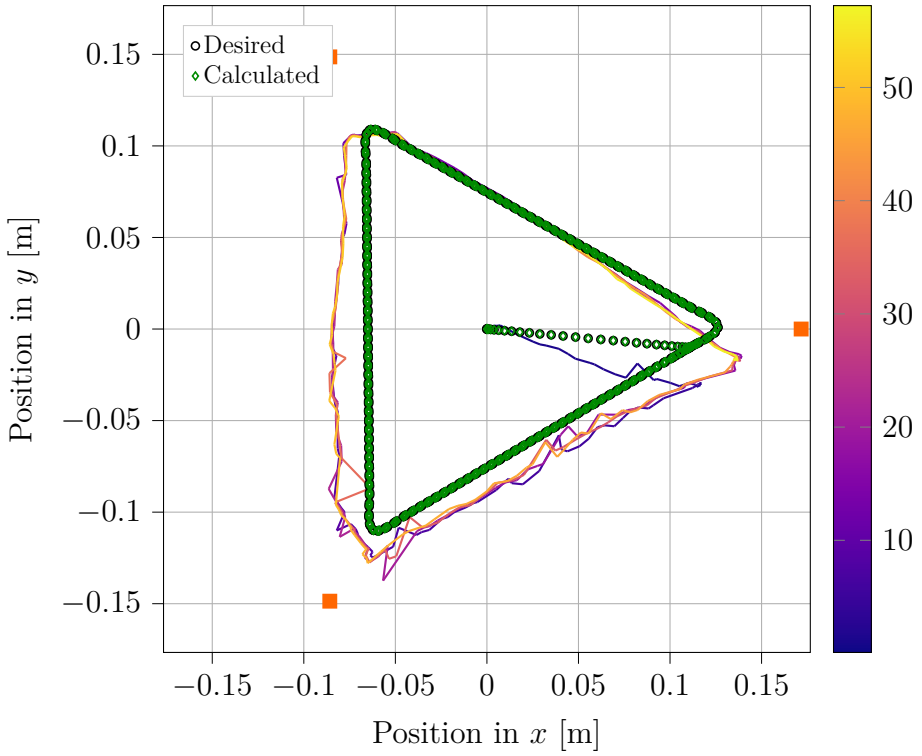


Figure 6.31: Triangular trajectory with sharp corners for $\Delta t = 0.1\,\mathrm{s}$ and $v_{\mathrm{lim}} = 0.05\,\frac{\mathrm{m}}{\mathrm{s}}$, measured (color gradient), calculated (green) and desired (black).

The results show once more, that both the step size Δt and the desired velocity v_{lim} have no significant effect on the robot's capability on tracking the triangular trajectory. Again, a general offset is seen for the straight paths in between

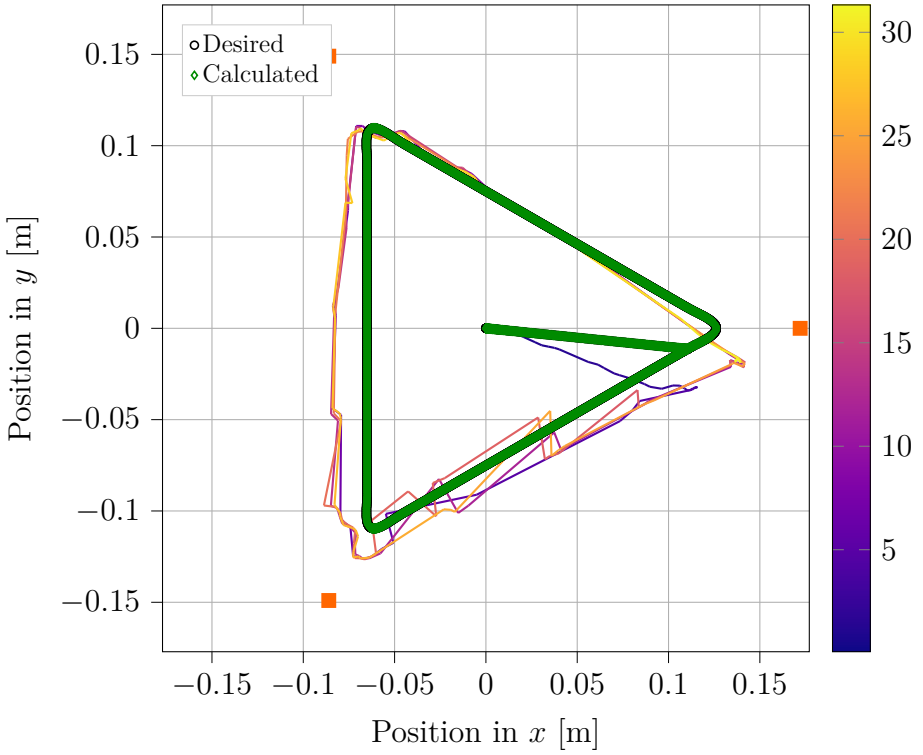


Figure 6.32: Triangular trajectory with sharp corners for $\Delta t = 0.01\,\mathrm{s}$ and $v_{\mathrm{lim}} = 0.1\,\frac{\mathrm{m}}{\mathrm{s}}$, measured (color gradient), calculated (green) and desired (black).

the motors 1 and 2 and motors 2 and 3. But the robot successfully tracks the sharp corners. Only close to the second motor the robot fails to follow the sharp corner. The robot struggles to perform the sharp turn and oscillating behavior occurs due to motor 2 working at its limit. Close to motor 1 the robot manages to represent the sharp corner, but again a clear offset is noticeable, which occurs due to the less accurate tuning of the parameters c_1 and b_1.

Finally, the sharp trajectory is considered again but rotated by 60°. The corners are located in between the motors now. All other parameters stay the same. Again no significant differences occur for different step sizes Δt and desired velocities v_{lim}. But the inaccuracies in tuning become more evident. The robot does not manage to reach the left corner between motor 2 and 3. At the opposite site the robot struggles with moving on the straight desired path and instead a small rounding occurs. This can be traced back to the inaccurate identification of the scaling parameters c_1 and b_1 once again, where motor 1 is pulling to strong. Contrarily, it applies too much load on the robot during the non-active

time, when the robot is pulled by the other two motors.

At last, the change in cable length at each time step is plotted in Fig. 6.35. The graph shows how during the initial deflection the motors 1 and 2 pull the robot, while motor 3 yields to allow bending. Afterwards, motor 1 moves in the opposite direction, when the robot reaches the first corner of the triangle between motor 2 and 3. Accordingly, motor two pulls stronger again and motor 3 moves in the opposite direction to bend the robot. Note the time shift of the input at each motor, first motor 1 moves in negative direction when being the non-active actuator (4.7 s-7.3 s), afterwards motor 2 does the likewise (7 s-10.5 s), and at last motor 3, where the robot reaches its starting position again (9 s-10.5 s). At those times the cable length induced by the virtual spring becomes 0, as the motor is not active.

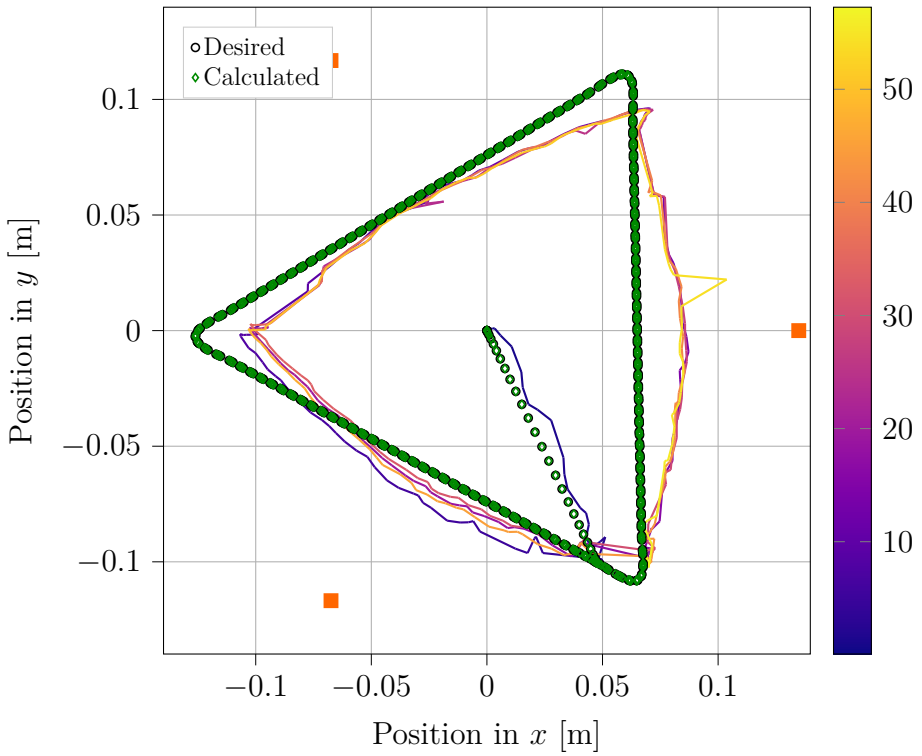


Figure 6.33: Triangular trajectory with sharp corners for $\Delta t = 0.1\,\mathrm{s}$ and $v_{\mathrm{lim}} = 0.05\,\frac{\mathrm{m}}{\mathrm{s}}$ but rotated by 60°, measured (color gradient), calculated (green) and desired (black).

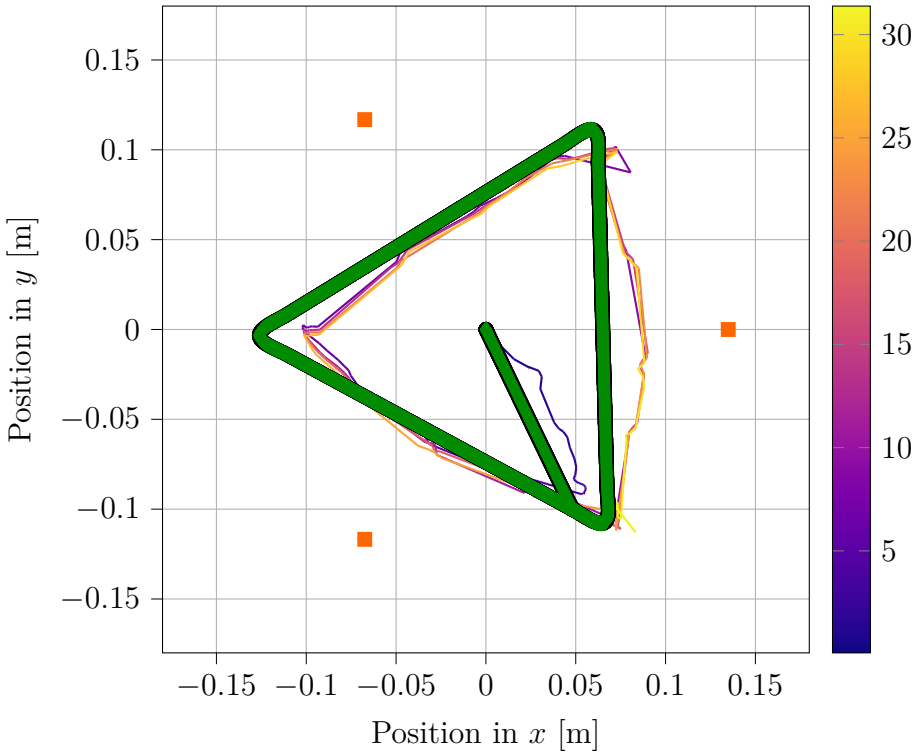


Figure 6.34: Triangular trajectory with sharp corners for $\Delta t = 0.01\,\mathrm{s}$ and $v_{\mathrm{lim}} = 0.1\,\frac{\mathrm{m}}{\mathrm{s}}$ but rotated by 60°, measured (color gradient), calculated (green) and desired (black).

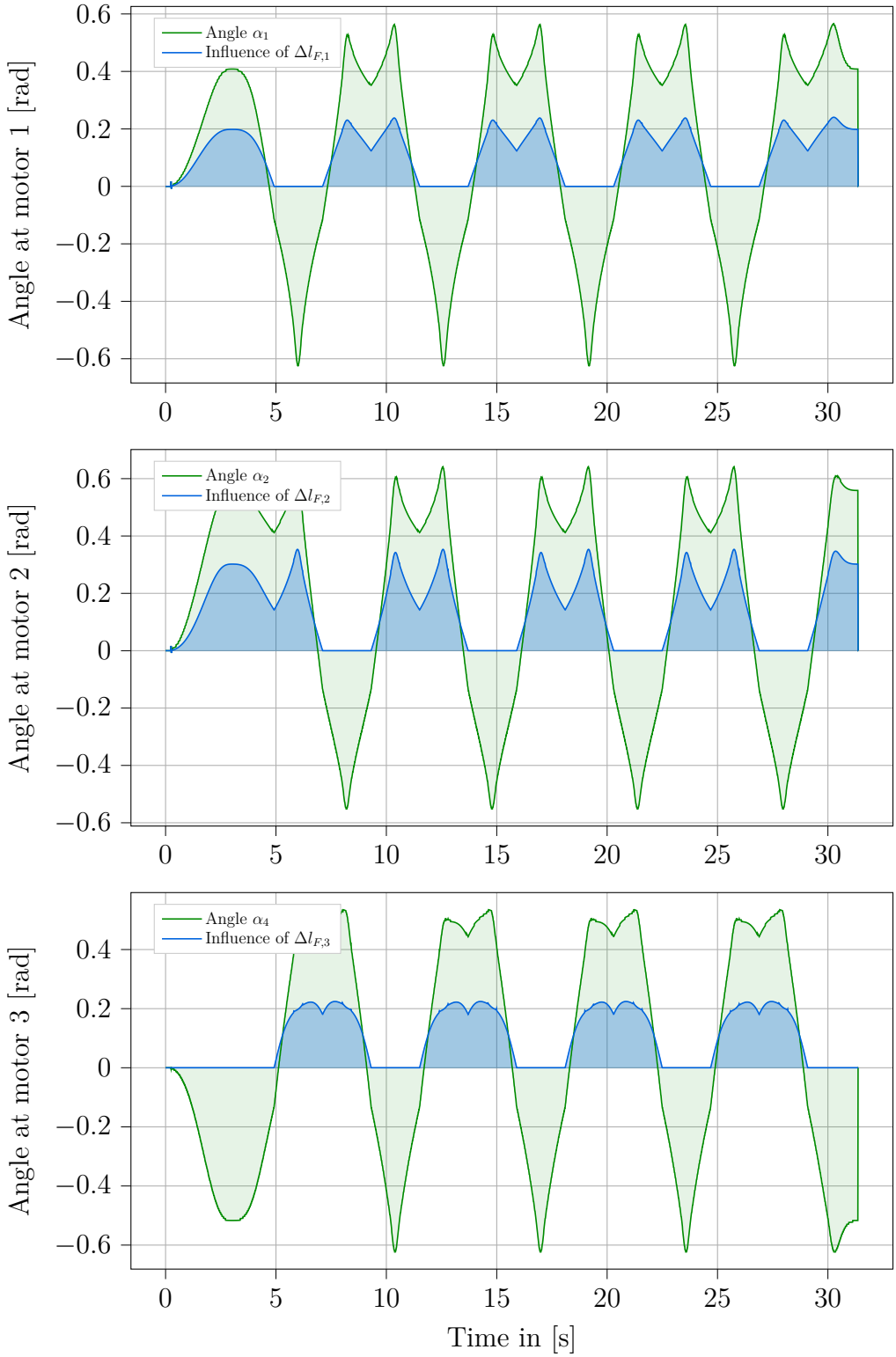


Figure 6.35: Motor angles α_q for $q = 1 \ldots 3$ over time t for the trajectory with sharp corners for $\Delta t = 0.01\,\mathrm{s}$ and $v_{\mathrm{lim}} = 0.1\,\frac{\mathrm{m}}{\mathrm{s}}$ but rotated by 60°.

Chapter 7

Summary and Outlook

In this work, the inverse model for a soft robot under consideration of the system dynamics is established. Forward kinematics is modeled and the equations of motion are derived to represent the forward dynamics. The servo-constraints approach is applied to derive an inverse model and solve the resulting DAE's. The inverse model provides a control law for feedforward control of the soft robot. In terms of modeling, the PCC-approach provides accurate modeling of the kinematics, with very small positional errors during slow motion. Furthermore, the application of the servo-constraints approach successfully solves the inverse problem, taking the inverse dynamics into account.

Furthermore, a new soft robot was manufactured. In a series of experiments, unknown parameters of the robotic system are identified and tuned in order to reach desired positions in space. Different strategies are introduced and studied to find suitable values for the parameters. Afterwards, trajectory following with variations of system parameters is evaluated. The effects of these parameters on the robots performance are examined. For any combination the system remains stable and the robot is able to follow desired trajectories in space, such as simple circular paths and more difficult triangular trajectories with sharper edges. However, the examined strategies of determining the parameters turns out to be insufficient. Constant values and values scaled by radius show satisfying results for larger trajectories, but following paths close to the robot's center is troublesome.

The experiments validate that trajectory following using the constructed soft robot is possible. However, inaccuracies during mounting of the system components and limitations towards hardware affect the results. Difference in initial strain on the length lead to one of the motors reaching it's limit and therefore struggling to follow trajectories of greater size. Additionally, during fast motion

the camera is not able to detect the robot's tip at all times and measurements are lost.

This motivates ideas for future work. First, the inverse model provides very good results but the manufacturing and mounting inaccuracies impair the results. It is expected that more accurate tracking can be achieved by installation of better hardware, such as more powerful actuators, and improvements in the tracking system, by increasing the speed of the tracking loop or choosing an alternative to the detection using AprilTags. Furthermore, actuation can be enhanced by finding a better representation for the non-linear actuation parameters. For example, neural networks can be applied to learn the parameters, which are expected to improve trajectory tracking, especially for paths close to the robot's center. Accurate trajectory following enables further research on control theory, for example by applying feedback control.

Bibliography

[AllenEtAl20] Allen, T.F.; Rupert, L.; Duggan, T.R.; Hein, G.; Albert, K.: Closed-form non-singular constant-curvature continuum manipulator kinematics. In 2020 3rd IEEE International Conference on Soft Robotics, pp. 410–416, 2020.

[AmouriMahfoudiZaatri19] Amouri, A.; Mahfoudi, C.; Zaatri, A.: Dynamic modeling of a spatial cable-driven continuum robot using euler-lagrange method. 2019.

[ArmaniniEtAl22] Armanini, C.; Boyer, F.; Mathew, A.T.; Duriez, C.; Renda, F.: Soft robots modeling: a structured overview, 2022.

[Bekman22] Bekman, T.: Model inversion of a soft robot for trajectory tracking, master thesis. Hamburg University of Technology, 2022.

[BlajerKolodziejczyk04] Blajer, W.; Kolodziejczyk, K.: A geometric approach to solving problems of control constraints: Theory and a dae framework. Multibody System Dynamics, pp. 343–364, 2004.

[BlajerSeifriedKolodziejczyk15] Blajer, W.; Seifried, R.; Kolodziejczyk, K.: Servo-constraint realization for underactuated mechanical systems. Archive of Applied Mechanics, 2015.

[CoevoetEtAl17] Coevoet, E.; Morales-Bieze, T.; Largilliere, F.; Zhang, Z.; Thieffry, M.; Sanz-Lopez, M.; Carrez, B.; Marchal, D.; Goury, O.; Dequidt, J.; Duriez, C.: Software toolkit for modeling, simulation, and control of soft robots. Advanced Robotics, pp. 1–17, 2017.

[Della SantinaEtAl20] Della Santina, C.; Katzschmann, R.K.; Bicchi, A.; Rus, D.: Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment. The International Journal of Robotics Research, 2020.

[DingEtAl22] Ding, L.; Niu, L.; Su, Y.; Yang, H.; Liu, G.; Gao, H.; Deng, Z.: Dynamic finite element modeling and simulation of soft robots. Chinese Journal of Mechanical Engineering, 2022.

[Drücker22] Drücker, S.: Servo-constraints for inversion of underactuated multibody systems. Hamburg University of Technology, 2022.

[FalkenhahnEtAl14] Falkenhahn, V.; Mahl, T.; Hildebrandt, A.; Neumann, R.; Sawodny, O.: Dynamic modeling of constant curvature continuum robots using the euler-lagrange formalism. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2428–2433, 2014.

[FangEtAl22] Fang, G.; Tian, Y.; Yang, Z.X.; Geraedts, J.M.P.; Wang, C.C.L.: Efficient jacobian-based inverse kinematics with sim-to-real transfer of soft robots by learning, 2022.

[Gent58] Gent, A.N.: On the relation between indentation hardness and young's modulus. Rubber Chemistry and Technology, Vol. 31, pp. 896–906, 1958.

[GiorelliEtAl12] Giorelli, M.; Renda, F.; Calisti, M.; Arienti, A.; Ferri, G.; Laschi, C.: A two dimensional inverse kinetics model of a cable driven manipulator inspired by the octopus arm. In 2012 IEEE International Conference on Robotics and Automation, pp. 3819–3824, 2012.

[GiorelliEtAl13] Giorelli, M.; Renda, F.; Ferri, G.; Laschi, C.: A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5033–5039, 2013.

[GouryDuriez18] Goury, O.; Duriez, C.: Fast, generic, and reliable control and simulation of soft robots using model order reduction. IEEE Transactions on Robotics, 2018.

[GrossEtAl09] Gross, D.; Hauger, W.; Schröder, J.; Wall, W.A.: Technische Mechanik 2: Band 2: Elastostatik (Springer-Lehrbuch). Springer, Berlin, 2009.

[GéradinCardona01] Géradin, M.; Cardona, A.: Flexible Multibody Dynamics: A Finite Element Approach. 2001.

[Itseez15] Itseez: Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[JensenEtAl22] Jensen, S.W.; Johnson, C.C.; Lindberg, A.M.; Killpack, M.D.: Tractable and intuitive dynamic model for soft robots via the recursive newton-euler algorithm. In 2022 IEEE 5th International Conference on Soft Robotics, pp. 416–422, 2022.

[Kirgetov67] Kirgetov, V.: The motion of controlled mechanical systems with prescribed constraints (servoconstraints). Journal of Applied Mathematics and Mechanics, pp. 465–477, 1967.

[LargilliereEtAl15] Largilliere, F.; Verona, V.; Coevoet, E.; Sanz-Lopez, M.; Dequidt, J.; Duriez, C.: Real-time control of soft-robots using asynchronous finite element modeling. In 2015 IEEE International Conference on Robotics and Automation, pp. 2550–2555, 2015.

[Mueller19] Mueller, A.: Modern robotics: Mechanics, planning, and control. IEEE Control Systems Magazine, pp. 100–102, 2019.

[Nystedt21] Nystedt, P.: Arc length of function graphs via taylor's formula. International Journal of Mathematical Education in Science and Technology, pp. 310–323, 2021.

[Olson11] Olson, E.: Apriltag: A robust and flexible visual fiducial system. pp. 3400 – 3407, 2011.

[PolygerinosEtAl15] Polygerinos, P.; Wang, Z.; Overvelde, J.T.B.; Galloway, K.C.; Wood, R.J.; Bertoldi, K.; Walsh, C.J.: Modeling of soft fiber-reinforced bending actuators. IEEE Transactions on Robotics, pp. 778–789, 2015.

[Powell64] Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal, pp. 155–162, 1964.

[RendaEtAl14] Renda, F.; Giorelli, M.; Calisti, M.; Cianchetti, M.; Laschi, C.: Dynamic model of a multibending soft robot arm driven by cables. IEEE Transactions on Robotics, pp. 1109–1122, 2014.

[RendaEtAl18] Renda, F.; Boyer, F.; Dias, J.; Seneviratne, L.: Discrete cosserat approach for multisection soft manipulator dynamics. IEEE Transactions on Robotics, pp. 1518–1533, 2018.

[RoneBen-Tzvi14] Rone, W.S.; Ben-Tzvi, P.: Continuum robot dynamics utilizing the principle of virtual power. IEEE Transactions on Robotics, pp. 275–287, 2014.

[RungeEtAl17] Runge, G.; Wiese, M.; günther, l.; Raatz, A.: A framework for the kinematic modeling of soft material robots combining finite element analysis and piecewise constant curvature kinematics. 2017.

[SchiehlenEberhard20] Schiehlen, W.; Eberhard, P.: Technische Dynamik, Aktuelle Modellierungs- und Berechnungsmethoden auf einer gemeinsamen Basis. Springer Vieweg Wiesbaden, 2020.

[Seifried14] Seifried, R.: Dynamics of Underactuated Multibody Systems: Modeling, Control and Optimal Design. 2014.

[ShamilyanEtAl23] Shamilyan, O.; Kabin, I.; Dyka, Z.; Sudakov, O.; Cherninskyi, A.; Brzozowski, M.; Langendoerfer, P.: Intelligence and motion models of continuum robots: An overview. IEEE Access, pp. 60988–61003, 2023.

[SicilianoEtAl08] Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G.: Robotics: Modelling, Planning and Control. Springer Publishing Company, Incorporated, 2008.

[SicilianoKhatib08] Siciliano, B.; Khatib, O.: Springer Handbook of Robotics. Springer Handbook of Robotics. Springer Berlin Heidelberg, 2008.

[SpillmannTeschner07] Spillmann, J.; Teschner, M.: Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. Eurographics, pp. 63–72, 2007.

[SullivanFlitzpatrick19] Sullivan, C.; Flitzpatrick, J.: Beyond the metal: Investigating soft robots at nasa langley. `https://www.nasa.gov/feature/langley/beyond-the-metal-investigating-soft-robots-at-nasa-langley`, 2019. Accessed: 2023-09-22.

[ThuruthelEtAl17] Thuruthel, T.; Falotico, E.; Renda, F.; Laschi, C.: Learning dynamic models for open loop predictive control of soft robotic manipulators. Bioinspiration & Biomimetics, 2017.

[ThuruthelEtAl18] Thuruthel, T.; Ansari, Y.; Falotico, E.; Laschi, C.: Control strategies for soft robotic manipulators: A survey. Soft Robotics, 2018.

[ThuruthelRendaIida20] Thuruthel, T.; Renda, F.; Iida, F.: First-order dynamic modeling and control of soft robots. Frontiers in Robotics and AI, Vol. 7, 2020.

[TillAloiRucker19] Till, J.; Aloi, V.; Rucker, D.: Real-time dynamics of soft and continuum robots based on at-rod models. The International Journal of Robotics Research, pp. 723–746, 2019.

[Tomlin01] Tomlin, C.: Nonlinear systems: analysis, stability, and control, shankar sastry, springer-verlag, new york, ny, 1999. International Journal of Robust and Nonlinear Control, pp. 789–793, 2001.

[TrivediLotfiRahn08] Trivedi, D.; Lotfi, A.; Rahn, C.D.: Geometrically exact models for soft robotic manipulators. IEEE Transactions on Robotics, pp. 773–780, 2008.

[WebsterJones10] Webster, R.; Jones, B.: Design and kinematic modeling of constant curvature continuum robots: A review. I. J. Robotic Res., pp. 1661–1683, 2010.

[Wiek21] Wiek, J.C.: Comparison of simulation methods for soft robots, project thesis. Hamburg University of Technology, 2021.

[Wiek22] Wiek, J.C.: Comparison of control methods for soft robots, master thesis. Hamburg University of Technology, 2022.

[Wood16] Wood, R.: squishy robot fingers aid deep sea exploration. `https://seas.harvard.edu/news/2016/01/squishy-robot-fingers-aid-deep-sea-exploration`, 2016. Accessed: 2023-09-22.

[WuEtAl21] Wu, S.; Ze, Q.; Dai, J.; Udipi, N.; Paulino, G.; Zhao, R.: Stretchable origami robotic arm with omnidirectional bending and twisting. Proceedings of the National Academy of Sciences, Vol. 118, p. e2110023118, 2021.

[XunZhengKruszewski23] Xun, L.; Zheng, G.; Kruszewski, A.: Cosserat-rod based dynamic modeling of soft slender robot interacting with environment, 2023.

[Zelinsky09] Zelinsky, A.: Trajectory planning for automatic machines and robots (biagiotti, l. et al; 2008). IEEE Robotics & Automation Magazine - IEEE ROBOT AUTOMAT, pp. 101–101, 2009.

[ZhengLin22] Zheng, T.; Lin, H.: Pde-based dynamic control and estimation of soft robotic arms, 2022.

Appendix

A.1 Contents Archive

There is a folder **MSC_059_Maroofi/** in the archive. The main folder contains the entries

- **MSC_059_Maroofi.pdf**: the pdf-file of the thesis MSC-059.
- **Data/**: a folder with all the relevant data, programs, scripts and simulation environments.
- **Latex/**: a folder with the *.tex documents of the thesis MSC-059 written in Latex and all figures (also in *.svg data format if available).
- **Presentation/**: a folder with the relevant data for the presentation including the presentation itself, figures and videos.

Erklärung

Ich, Sean Maroofi (Student des Maschinenbaus an der Technischen Universität Hamburg, Matrikelnummer 51334), versichere, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Unterschrift	Datum