Modeling, Sensing, and Control for Agile Trajectory Tracking of Soft Robots

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation (Monografie)

von
Malte Grube

aus
Zeven

2026

1. Gutachter: Prof. Dr.-Ing. Robert Seifried
2. Gutachter: Prof. Dr.-Ing. Kristin de Payrebrune

Tag der mündlichen Prüfung: 09. 12. 2025

MuM Notes in Mechanics and Dynamics

Editor: Prof. Dr.-Ing. Robert Seifried
Hamburg University of Technology
Institute of Mechanics and Ocean Engineering (MuM)
www.tuhh.de/mum

Volume 13
Malte Grube
“Modeling, Sensing, and Control for Agile Trajectory Tracking of Soft Robots”
Hamburg, 2026

DOI: https://doi.org/10.15480/882.16343
ORCID: https://orcid.org/0000-0002-3882-537X

Acknowledgments

This work results from research conducted at the Institute of Mechanics and Ocean Engineering (MuM) at Hamburg University of Technology. I would like to thank everyone who has contributed to this work in one way or another.

First of all, I would like to thank my supervisor, Prof. Robert Seifried, for his continuous guidance and support. I am particularly grateful for the freedom he has given me to try my hand at research and teaching, and for the trust he has placed in me in doing so. Next, I would like to thank Prof. Kristin de Payrebrune for acting as the second reviewer of this work, as well as Prof. Jürgen Grabe for chairing the examination committee.

I would also like to thank my colleagues for the wonderful time at MuM. Thank you for the countless discussions about research, teaching, and everything else. And thank you for all the activities around: The cake Mondays, the countless barbecues, the canoe tours with our exchange students and all the other events. This is what made time at MuM special.

I would also like to thank our technical assistants, Wolfgang, Riza, and Norbert, as well as our senior engineer, Marc-André, for their help and ideas in conducting lab experiments.

Finally, I would like to thank my family for their unconditional support and encouragement, and for always standing by me and supporting me wherever possible. Thank you.

Malte Grube

Grabau, January 2026

To my family.

Abstract

For modern soft robotic applications, accurate agile control is important. This requires models that accurately represent both quasi-static and dynamic behavior, soft sensors for state estimation, and dynamic controllers. In this work, modeling approaches for soft robots are first presented and evaluated in terms of their accuracy and efficiency. Model-based approaches are effective in modeling dynamics, while the strength of data-driven approaches lies in modeling quasi-static behavior, including hard-to-model effects such as manufacturing inaccuracies. Second, curvature sensors for integration into the soft robot body to measure the configuration of the soft robot are presented. The application to shape estimation for soft robot grippers and to tip position estimation is experimentally demonstrated. Third, control approaches for soft robots are presented and evaluated in simulation and experiment. Feedforward controllers are important for soft robots to keep the sensor requirements low, feedback controllers are needed for disturbance rejection. This thesis covers approaches from all these areas to enable agile trajectory tracking control for soft robots and expand their application range.

Contents

List of Figures

List of Tables

Introduction

Recent growth in robotics is increasingly shifting to new applications. Some of the most important areas for future growth in robotics are healthcare, advanced manufacturing, and logistics. In healthcare, robots are being developed for care, rehabilitation, and minimally invasive surgery [Riek17]. In advanced manufacturing, deeper integration of robots into the production process, even for smaller lot sizes, is becoming more important, requiring human-machine interaction and flexible robots [KeshvarparastEtAl24]. In logistics, the handling of fragile and variably shaped objects, as well as interaction with unstructured environments, is becoming increasingly important, for example to enable fully automated packaging in warehouses [FerreiraReis23]. In all of these applications, however, the robot requirements profile is very different from that of traditional industrial applications. Traditional applications typically require high stiffness, high absolute accuracy and high strength. This typically comes at the cost of low flexibility, specialization for a very specific task, high energy consumption, and high risk potential. For modern robotic applications, however, low risk potential to enable close collaboration and high flexibility are important [KeshvarparastEtAl24].

One answer to these new challenges in robotics is the development of soft robots. Soft robots are robots typically made of soft material such as silicone or foam that can undergo large deformations and rely on inherent or structural compliance [López-GonzálezEtAl23]. This typically results in a very high or unlimited number of degrees of freedom [LeeEtAl17a]. Due to their softness, they are usually inherently safe and therefore well suited for human-machine interaction. In addition, their softness allows them to adapt to the environment. While softness and flexibility open up new possibilities in robotics, they also bring new engineering challenges, as many methods and components established in conventional robotics cannot be directly transferred to soft robotics. In particular, new concepts for actuation, sensor integration, modeling, and control are the scope of current research [LeeEtAl17a].

1.1 Background and Motivation

One of the main obstacles for a wider application of soft robots is their typically complex dynamic behavior, which mostly limits their use to applications where slow motions can be accepted. Several main components are needed to overcome this challenge to enable agile trajectory tracking control. These components are accurate and computationally efficient models that represent the quasi-static and dynamic behavior of soft robots, sensors for state estimation that can be fully integrated into soft robots, and control algorithms that match the specific needs of soft robots.

The main challenges in modeling soft robots are the large elastic deformations that occur and the parameter uncertainties that result from manufacturing inaccuracies and unknown or changing material parameters. The two main types of models used in soft robotics are beam models and data-driven models [ArmaniniEtAl23]. Most soft robots are beam-shaped or consist of several beam-shaped components. Bending is usually the dominant deformation. Beam models are a simple and efficient way to model this type of robot. However, effects such as manufacturing inaccuracies are difficult to incorporate into the model. In addition, more complex shaped robots require different models. An alternative is data-driven models. They can be trained directly on measurements of real soft robot motions and therefore allow the inclusion of properties that are otherwise difficult to model [KimEtAl21]. Various supervised and reinforcement learning techniques are used in soft robotics [WangEtAl21]. However, data-driven modeling comes at the cost of reduced insight into the model and potentially high training data requirements, especially if not only quasi-static but also dynamic behavior is to be included in the model.

The second component considered is soft sensors for state estimation of soft robots. Sensor feedback is an important component to achieve accurate control of soft robots, especially in the presence of disturbances [LeeEtAl17a]. Due to the softness and the typically large strains that occur, sensor integration into soft robots is often difficult. Conventional rigid or flexible sensors usually cannot be used as they either compromise the softness of the soft robot or cannot withstand the large strains that occur in soft robots [RusTolley15]. In practical applications, external sensors such as camera tracking systems, which are popular in research, are often not an option because they limit the flexibility of the soft robot system. Thus, soft sensors that can be integrated into the body of the soft robot are important. Therefore, various resistive, capacitive, inductive and piezoelectric sensors are the scope of current research [LeeEtAl17a]. For soft robots, bending is usually the dominant deformation and therefore an estimation of curvature is necessary. Moreover, one of the most popular modeling techniques for soft robots, the piecewise constant curvature (PCC) model, is based on a description of bending. This makes curvature sensors particularly important for soft robots.

The third component considered is different control concepts for agile trajectory tracking control. A distinction is made between feedforward and feedback control. Since sensor integration in soft robots is difficult and therefore often only limited or no sensor feedback is available, feedforward control is particularly important in soft robotics [LeeEtAl17a]. However, perturbations typically occur in real applications. This also makes a feedback component essential. A combination of feedforward and feedback controllers can help to reduce the amount and quality of sensor data required, while still being able to respond to environmental influences. Additionally, a distinction is made between quasi-static and dynamic controllers. Quasi-static controllers are popular and widely used because of their simplicity; however, they can only be applied if slow motions can be tolerated to prevent the excitation of large oscillations. For applications requiring more agility, dynamic controllers are needed. Challenges in the control of soft robots arise from their typically large and continuous deformations and actuators that are distributed throughout the soft robot body. This often results in complex models with many degrees of freedom, which are often underactuated and redundantly actuated and therefore difficult to control [Della SantinaEtAl23].

1.2 Aim and Structure of this work

This work aims to extend the applicability of soft robots to more dynamic applications. For this purpose, mechanical and data-driven models, sensors for state estimation, and feedforward and feedback control concepts, which are the main components required for agile trajectory tracking control of soft robots, are developed and investigated in simulation and experiment.

In chapter 2 several dynamic beam models of soft robots for large deformations are presented and evaluated with respect to their accuracy and computational efficiency. The first model presented is the piecewise constant curvature model (PCC), which is one of the simplest and most widely used models in soft robotics. As a second model, the Cosserat rod model is considered, which provides a more accurate representation of the beam mechanics. As a third model, the absolute nodal coordinate formulation (ANCF), which is well known from multibody dynamics, is presented. Finally, the integration of tendon actuation, which is used throughout this thesis, into the beam models is described. The models presented here form the basis for simulation and especially the model-based controller design in the following chapters.

In chapter 3 the data-driven modeling of soft robots is investigated. In a first step, a soft robot test system is introduced, which is used for experimental evaluation

throughout this thesis. In a second step, a feedforward neural network based quasi-static model for this test system is presented. In a third step, a dynamic data-driven model based on spectral submanifold reduction (SSMR) is presented. The accuracy of both models is experimentally evaluated on the soft robotic test system in terms of accuracy and training effort. The quasi-static model is used throughout this work as a basis for controller design.

In chapter 4 a planar and a spatial soft curvature sensor for direct integration into soft robots are presented. First, a planar curvature sensor is presented. The curvature sensor can determine the magnitude of the curvature in two spatial directions, but not the direction, which limits its use mainly to planar applications. Based on this sensor, a spatial curvature sensor is presented that can determine both the magnitude and the direction of curvature in two spatial directions. This sensor can be applied to state estimation of soft robots undergoing spatial deformations. For both sensors, the design and fabrication are described, followed by an experimental evaluation of the sensor accuracy and an experimental investigation of the application to soft robots.

In chapter 5 different concepts for quasi-static and dynamic feedforward control are presented. A data-driven quasi-static, a model-based dynamic and a hybrid dynamic controller are considered. The hybrid controller consists of a data-driven inverse quasi-static model and a model-based inverse dynamic model based on servo constraints. This allows to combine the strengths of data-driven and model-based controllers in terms of accuracy and data efficiency. Finally, the three controllers are compared experimentally in terms of accuracy and computational efficiency on the test system.

In chapter 6 different feedback control concepts are presented and compared. In a first step, basic combined feedforward and feedback control approaches are applied to a physical test system with an integrated curvature sensor for feedback and tested in experiment. In a second step, more advanced quasi-static and dynamic feedback control concepts are presented and compared in simulation regarding their accuracy, computational efficiency and robustness to parameter uncertainty. A learning-based quasi-static controller, a linear quadratic regulator with integral action (LQI) and gain scheduling and a model predictive controller (MPC) are considered.

Finally, in chapter 7 the work is summarized and an outlook on future research directions is given.

Beam Models

Many soft robots are beam-shaped, or composed of multiple beam-shaped components, since beams naturally allow for large, continuous deformation, which is an essential characteristic of most soft robots. This makes beam models, along with data-driven models discussed in chapter 3, one of the most popular models used in soft robotics. Since deformations in soft robotics are typically large, standard geometrically linear beam models such as the Euler-Bernoulli beam or the Timoschenko beam cannot usually be applied. Instead, more complex beam models such as Kirchhoff-Love [ColemanEtAl93] or Cosserat [Antman74] rod models are used, see e.g. [TrivediEtAl08, RuckerWebster III11, RendaEtAl14, BlackEtAl18, Janabi-SharifiEtAl21] among others. Cosserat models are by far more widely used in the soft robotics community than Kirchhoff-Love models. One of the most popular beam models used in soft robotics is the piecewise constant curvature (PCC) model [WebsterJones10], which can be considered as a simplification of the Cosserat model or as an extension of the finite segment model. Besides these models, other beam models such as the Absolute Nodal Coordinate Formulation (ANCF) [Shabana11], which is a well-known method from multibody dynamics, can be used to model large deformations of beam-shaped soft robots.

In this chapter, first in Secs. 2.1 to 2.3, the piecewise constant curvature model, the Cosserat model and the absolute nodal coordinate formulation are presented. Afterwards in Secs. 2.4 and 2.5, these models are compared regarding their accuracy and computational efficiency for typical soft robotic problems. Finally in Sec. 2.6, the integration of tendon actuation, which is one of the most popular actuation methods in soft robotics, into the simulation models is described. First results have been presented in [Wieck21].

2.1 Piecewise Constant Curvature

The piecewise constant curvature (PCC) model [WebsterJones10] is one of the simplest and most intuitive models for beams undergoing large deformations

used in soft robotics. In the following, the derivation of the PCC model loosely follows [RoneBen-Tzvi14]. The PCC model discretizes a beam into multiple pieces of constant curvature using a 1D finite element (FE) mesh. For many soft robots, the constant curvature assumption is a very good approximation. In particular, tendon actuation and fluidic actuation of soft robots often lead to deformations with constant curvature [WebsterJones10]. On the other hand, external loads such as gravity usually lead to deformations with non-constant curvature. As a result, PCC models often allow very efficient modeling with discretization into just a few pieces, as long as the external loads are not too large. Bending in two spatial directions, torsion and strain can be modeled with the PCC model. Shear is neglected. The elongation mode is usually very stiff compared to the bending and torsion modes. Therefore, it is often neglected, as done in the following. Torsion is also often neglected because in many applications there are no or only very small torsional moments and therefore the torsion is often very small. This is also the case for the test system introduced in Sec. 3.1 and used throughout this work.

2.1.1 Spatial Discretization

In Fig. 2.1 the spatial PCC discretization into N_{pieces} pieces is shown. Each piece consists of a massless elastic link at the end of which a mass-loaded disk is mounted. The stiffness and damping properties are aggregated in the elastic link, and the inertia properties are lumped in the disk. The deformation of each piece with the index $i = 1 \dots N_{\text{pieces}}$ is described by the curvatures around the $x_{i-1}-$ and $y_{i-1}-$ axis γ_i and β_i and by the torsion ε_i around the $z_{i-1}-$axis. At the tip of each piece the coordinate system $\mathrm{K}_i : \{\mathrm{O}_i, x_i, y_i, z_i\}$ is attached, at the bottom of each piece the coordinate system $\mathrm{K}_{i-1} : \{\mathrm{O}_{i-1}, x_{i-1}, y_{i-1}, z_{i-1}\}$ is attached.

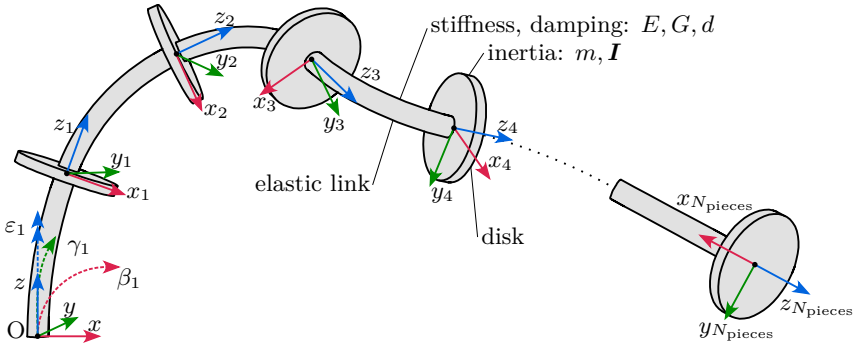


Figure 2.1: PCC discretization of a beam-shaped soft robot into N_{pieces} pieces.

For simplicity, it is assumed, that the coordinate system $\mathrm{K}_0 : \{\mathrm{O}_0, x_0, y_0, z_0\}$ attached at the beginning of the first piece coincides with the global coordinate system $\mathrm{K} : \{\mathrm{O}, x, y, z\}$.

2.1.2 Internal Forces and Moments

For linear elastic material behavior and viscous damping the internal moments $\boldsymbol{\ell}_{i,\mathrm{loc}}$ in the local coordinate frame $\mathrm{K}_{\mathrm{i}-1}$ of each segment result in

$$\boldsymbol{\ell}_{i,\mathrm{loc}} = \boldsymbol{C} \begin{bmatrix} -\beta_i \\ \gamma_i \\ \varepsilon_i \end{bmatrix} + \boldsymbol{D} \begin{bmatrix} -\dot{\beta}_i \\ \dot{\gamma}_i \\ \dot{\varepsilon}_i \end{bmatrix} \tag{2.1}$$

with the stiffness matrix

$$\boldsymbol{C} = \begin{bmatrix} EI_1 & & \\ & EI_2 & \\ & & GJ \end{bmatrix}. \tag{2.2}$$

Here E is the Young's modulus of the beam, G is the shear modulus, I_1 and I_2 are the second moments of area about the x- and y-axis and J is the polar second moment of area. The damping matrix

$$\boldsymbol{D} = \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{bmatrix} \tag{2.3}$$

can usually only be determined experimentally. For simplicity, a diagonal matrix structure is assumed. Since in this work elongation is not included in the model, there are no internal forces $\boldsymbol{f}_i$.

2.1.3 Equations of Motion

The equations of motion can be derived using the Newton-Euler algorithm [SchiehlenEberhard20, ch. 5]. In a first step, the position $\boldsymbol{p}_i$ and rotation $\boldsymbol{R}_i$ of the disks in the global coordinate frame is derived. The quantities required to derive the position vector are visualized in Fig. 2.2. The position $\boldsymbol{p}_{i,\mathrm{loc}}$ of the disk center of the i-th piece in local coordinates is completely described by the curvatures β_i and γ_i. It is defined as

$$\boldsymbol{p}_{i,\mathrm{loc}} = \begin{bmatrix} \cos(\varphi_i) \cdot \frac{1-\cos(\theta_i)}{k_i} \\ \sin(\varphi_i) \cdot \frac{1-\cos(\theta_i)}{k_i} \\ \frac{\sin(\theta_i)}{k_i} \end{bmatrix} \tag{2.4}$$

with the bending angle

$$\theta_i = k_i \cdot \ell_i, \tag{2.5}$$

the angle of the bending plane

$$\varphi_i = \arctan\left(\frac{\gamma_i}{\beta_i}\right), \tag{2.6}$$

see Fig. 2.2, and the magnitude of the total curvature

$$k_i = \sqrt{\beta_i^2 + \gamma_i^2}. \tag{2.7}$$

The variable ℓ_i denotes the length of piece i.

The rotation matrix $\boldsymbol{R}_{\mathrm{K}_{i-1}\mathrm{K}_i}$, which describes the rotation from disk i to disk $i-1$, can be determined from a concatenation of three elementary rotations. First there is a rotation around the z_{i-1}-axis by the angle φ_i into the bending plane. This is followed by a rotation around the new y-axis by the bending angle θ_i. Finally, there is a rotation around the new z-axis by the angle $(\varepsilon_i - \varphi_i)$. Here ε_i is the contribution of the torsion and $-\varphi_i$ rotates back from the bending plane.

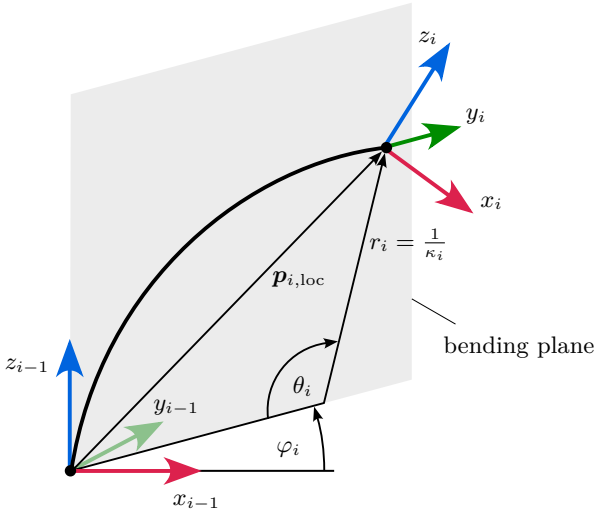


Figure 2.2: Description of the deformation of a single piece with the PCC model.

This results in

$$\begin{aligned}\boldsymbol{R}_{\mathrm{K}_{j-1}\mathrm{K}_j} = &\begin{bmatrix}\cos(\varphi_i) & -\sin(\varphi_i) & 0\\ \sin(\varphi_i) & \cos(\varphi_i) & 0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}\cos(\theta_i) & 0 & \sin(\theta_i)\\ 0 & 1 & 0\\ -\sin(\theta_i) & 0 & \cos(\theta_i)\end{bmatrix}\\ &\cdot\begin{bmatrix}\cos(\varepsilon_i-\varphi_i) & -\sin(\varepsilon_i-\varphi_i) & 0\\ \sin(\varepsilon_i-\varphi_i) & \cos(\varepsilon_i-\varphi_i) & 0\\ 0 & 0 & 1\end{bmatrix}.\end{aligned} \tag{2.8}$$

The rotation from the coordinate frame K_i to the global coordinate frame K of disk i is described by the rotation matrix $\boldsymbol{R}_{\mathrm{KK}_i}$. It is calculated from the product of the rotation matrices between successive pieces as follows

$$\boldsymbol{R}_{\mathrm{KK}_i} = \prod_{j=1}^{i} \boldsymbol{R}_{\mathrm{K}_{j-1}\mathrm{K}_j}. \tag{2.9}$$

The global position vector is obtained by transforming equation (2.4) into the reference coordinate system using the rotation matrix $\boldsymbol{R}_{\mathrm{KK}_{i-1}}$ and adding the position vector of the previous piece $\boldsymbol{p}_{i-1}$. This results in

$$\boldsymbol{p}_i = \begin{cases}\boldsymbol{p}_{i,\mathrm{loc}} & i=1\\ \boldsymbol{p}_{i-1} + \boldsymbol{R}_{\mathrm{K}_{i-1}\mathrm{K}_i}\cdot\boldsymbol{p}_{i,\mathrm{loc}} & i>1\end{cases}. \tag{2.10}$$

In addition to the position vectors and rotation matrices, the applied forces $\boldsymbol{f}_i^{\mathrm{a}}$ and moments $\boldsymbol{\ell}_i^{\mathrm{a}}$ must be determined. They result in

$$\boldsymbol{f}_i^{\mathrm{a}} = \hat{\boldsymbol{f}}_i, \tag{2.11}$$

$$\boldsymbol{\ell}_i^{\mathrm{a}} = \boldsymbol{\ell}_i + \hat{\boldsymbol{\ell}}_i. \tag{2.12}$$

Here, $\boldsymbol{\ell}_i$ are the internal moments in the global coordinate frame, which can be calculated from the internal moments $\boldsymbol{\ell}_{i,\mathrm{loc}}$ in the local coordinate frame from equation (2.1) as

$$\boldsymbol{\ell}_i = \boldsymbol{R}_{\mathrm{KK}_i}\boldsymbol{\ell}_{i,\mathrm{loc}}, \tag{2.13}$$

and $\hat{\boldsymbol{f}}_i$ and $\hat{\boldsymbol{\ell}}_i$ are external forces and moments resulting e.g. from gravity or actuation.

With the position vectors and rotation matrices from equations (2.9) and (2.10) as well as the moments and forces from equations (2.11) and (2.12), the equations of motion can be derived as a function of the generalized variables using the Newton-Euler formalism. A straightforward choice for the generalized variables is the vector

$$\boldsymbol{y} = \begin{bmatrix}\beta_1 & \cdots & \beta_{N_{\mathrm{pieces}}} & \gamma_1 & \cdots & \gamma_{N_{\mathrm{pieces}}} & \varepsilon_1 & \cdots & \varepsilon_{N_{\mathrm{pieces}}}\end{bmatrix}^{\top}. \tag{2.14}$$

The equations of motion can be written as a system of $3N_{\text{pieces}}$ coupled second-order nonlinear differential equations of the form

$$\boldsymbol{M}(\boldsymbol{y},t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) = \boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t). \tag{2.15}$$

Here $\dot{\boldsymbol{y}}$ and $\ddot{\boldsymbol{y}}$ are the first and second derivatives of the vector of generalized coordinates $\boldsymbol{y}$ given by equation (2.14) and t is time. Further, $\boldsymbol{M}(\boldsymbol{y},t)$ is the time- and state-dependent mass matrix, $\boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t)$ is the vector of generalized Coriolis, centrifugal, and gyroscopic forces, and $\boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t)$ is the vector of generalized applied forces and moments.

2.1.4 A Note on Singularities and Alternative Parametrizations

One drawback of PCC models is the occurrence of removable singularities in many existing formulations, particularly when the segment is in a straight configuration, i.e. $\beta_i = \gamma_i = k_i = 0$. This singularity is not a system property, but results from the chosen parametrization. In the following, first the singular terms occurring in the formulation presented above are discussed. Then, some strategies to treat the singularities in practical applications are presented.

Singular Terms

In the parameterization introduced earlier with the state vector (2.14), the two terms

$$\frac{1-\cos(\theta_i)}{k_i}, \tag{2.16}$$

$$\frac{\sin(\theta_i)}{k_i}, \tag{2.17}$$

which occur in the position vector $\boldsymbol{p}_i$ defined in equation (2.4), become singular when the curvature k_i approaches zero, corresponding to the straight configuration. However, by applying L'Hospital's rule, it can be shown that the limit values of the terms (2.16) and (2.17) exist and result in

$$\lim_{k_i\to 0} \frac{1-\cos(\theta_i)}{k_i} = 0, \tag{2.18}$$

$$\lim_{k_i\to 0} \frac{\sin(\theta_i)}{k_i} = 1. \tag{2.19}$$

This allows the singular terms (2.16) and (2.17) to be replaced with their continuous extensions, ensuring smooth and differentiable behavior of the model [Steinmetz24, ch. 12].

Treating the Singularities

To address the issue of the singularities, several approaches have been proposed in the literature, some of which are discussed in the following. The simplest way to avoid problems caused by these singularities is to exclude the straight configuration from the workspace. This can be done by planning the trajectories so that none of the pieces reach the straight configuration. In many practical applications, this strategy is sufficient and avoids the need for more complex treatment of the singularities. This approach is used throughout this chapter. However, it should be noted that if the application inherently requires frequent near-straight configurations, trajectory design becomes significantly more complex and often impractical.

Alternatively, straight pieces can be approximated by slightly curved ones during simulation, as suggested in [RoneBen-Tzvi14]. This concept works well for quasi-static forward simulations, but easily runs into numerical problems for dynamic applications and inverse problems as the model becomes discontinuous.

Another approach, suggested e.g. in [DianEtAl22], is to replace the singularities by their Taylor series expansion. For the formulation of the PCC model presented here, this leads to replacing the singular terms (2.16), (2.17) by their Taylor series expansion around $k_i = 0$. This results in a singularity-free model that is continuous and differentiable. However, since the position vector $\boldsymbol{p}_i$ is used to derive the equations of motion, the singularities occur not only in the position vector $\boldsymbol{p}_i$ but propagate throughout the resulting equations of motion. Consequently, systematically replacing these terms throughout the model can become tedious, especially for large models.

As a fourth approach, a wide range of different formulations of PCC models have been proposed in the literature. For example, the formulation presented in [AllenEtAl20] allows a simpler treatment of singularities, since fewer singular terms appear in the equations of motion. Other formulations, such as [Della SantinaEtAl20a], avoid singularities altogether through a clever choice of parameterization, which is less intuitive. For a broader overview of PCC parameterizations, the reader is referred to [WebsterJones10, Della SantinaEtAl20a, Della SantinaEtAl23].

2.2 Cosserat Rod

The Cosserat rod (CR) model [Antman74] is a geometrically exact model for simulating rods. It has recently gained popularity in the soft robotics community because it is more accurate than the PCC model, which is the most popular model in soft robotics. The CR model can be considered as a geometrically nonlinear extension of the Timoshenko beam. It allows the modeling of large bending, torsion, elongation, and shear deformations. In the following, the description of the CR model follows [LangEtAl11].

In the continuous case, the kinematics of the beam can be completely described by

$$\boldsymbol{x}(s,t) : [0,L] \times [0,T] \to \mathbb{R}^3, \tag{2.20}$$

$$\boldsymbol{p}(s,t) : [0,L] \times [0,T] \to \mathbb{S}^3, \tag{2.21}$$

with

$$\mathbb{S}^3 = \{\boldsymbol{p} \in \mathbb{H} : ||\boldsymbol{p}|| = 1\}. \tag{2.22}$$

Here $\boldsymbol{x}(s,t)$ is the position of a point along the centerline at position s in the reference configuration at time t. The orientation of the beam cross-section at position s at time t is described by the unit quaternion $\boldsymbol{p}(s,t)$. It is assumed that the cross-sections remain plain. The parameter $s \in [0,L]$, which runs along the centerline of the beam, describes the position of a point in the undeformed reference configuration. Here, L is the total length of the undeformed centerline. The dynamics of the beam can then be described as a partial differential equation (PDE), see e.g. [LangEtAl11]. For practical applications, this PDE can usually only be solved numerically. Therefore, a discretization in space and time is required.

2.2.1 Spatial Discretization

For the spatial discretization of beams with large deformations, it is important to use a discretization scheme that preserves the objectivity of the strain measures when interpolating finite rotations. This requirement implies that the resulting deformations must remain invariant under rigid body motion. In general, for geometrically nonlinear finite element (FE) discretization schemes, this leads to complicated and usually numerically expensive interpolation schemes. Examples for FE formulations of Cosserat rods are found e.g. in [CrisfieldJelenic99, BetschSteinmann02, Sander10].

An alternative to FE formulations is the discretization on a staggered grid as presented in [LangEtAl11] and shown in Fig. 2.3 for a discretization of one rod

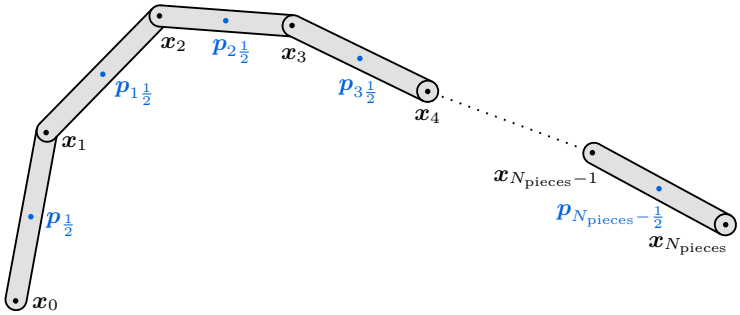


Figure 2.3: Staggered grid discretization of a beam-shaped soft robot into N_{pieces} pieces for the Cosserat rod model.

into N_{pieces} pieces. This discretization allows a simple interpolation scheme and will be used in the following. The translatory degrees of freedom are located at the vertices, the rotatory degrees of freedom are located at the segment midpoints as quaternions. In the following, the index notation $n = 0, 1, \ldots, N_{\text{pieces}}$ is used for quantities located on the vertices and the index notation $\nu = \frac{1}{2}, 1\frac{1}{2}, \ldots, N_{\text{pieces}} - \frac{1}{2}$ is used for quantities that are located on the midpoints between the two nodes with the indices $n-1$ and n.

2.2.2 Strain

To calculate the internal forces and moments, the bending, torsion, strain and shear of the beam must be calculated. The curvature and torsion of two elements n and $n-1$ with respect to each other are summarized in the vector of curvature and torsion $\boldsymbol{\kappa}_n$ which is assigned to the vertices. It can be calculated from the quaternions of two successive pieces $\boldsymbol{p}_{n-\frac{1}{2}}$ and $\boldsymbol{p}_{n+\frac{1}{2}}$ by

$$\boldsymbol{\kappa}_n = \frac{2}{l_n} \xi(\theta_n) \cdot \operatorname{Im}(\boldsymbol{w}_n) \tag{2.23}$$

with the angle between $\boldsymbol{p}_{n-\frac{1}{2}}$ and $\boldsymbol{p}_{n+\frac{1}{2}}$ in quaternionic space

$$\theta_n = \arccos \operatorname{Re}(\boldsymbol{w}_n), \tag{2.24}$$

the length of piece n

$$l_n = \begin{cases} \frac{L}{2} & n = 0 \vee n = N_{\text{pieces}} \\ L \end{cases} \tag{2.25}$$

and the quotient quaternion which describes the rotation between two successive pieces

$$\boldsymbol{w}_n = \bar{\boldsymbol{p}}_{n-\frac{1}{2}} \cdot \boldsymbol{p}_{n+\frac{1}{2}}. \tag{2.26}$$

$$\tag{2.27}$$

Where $\xi(\theta_n)$ is a weighting function, $\bar{\boldsymbol{p}}_{n-\frac{1}{2}}$ is a conjugate quaternion and L is the total length of the rod. In [LangEtAl11], based on [BauchauTrainelli03] different choices for the weighting function $\xi(\theta_n)$ are presented. Section 2.2.5 compares the effects of different weighting functions based on simulation results. One possible choice, which will be used in the following unless otherwise noted, is

$$\xi(\theta_n) = \sqrt{\frac{2}{1+\cos\theta_n}}. \tag{2.28}$$

The rate of change of curvature $\dot{\boldsymbol{\kappa}}_n$ required for the damping terms can be determined from the curvature vector $\boldsymbol{\kappa}_n$ by total differentiation with respect to time.

The strain and shear vector $\boldsymbol{\gamma}_n$ of the element n can be expressed as a function of the position of the end nodes $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$. It is assigned to the midpoint node $n-\frac{1}{2}$ of the element and calculated as

$$\boldsymbol{\gamma}_{n-\frac{1}{2}} = \boldsymbol{R}(\boldsymbol{p}_{n-\frac{1}{2}})^\top \cdot \frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1}}{L} - \begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top. \tag{2.29}$$

Here $\boldsymbol{R}(\boldsymbol{p}_{n-\frac{1}{2}})$ is the rotation matrix for the quaternion $\boldsymbol{p}_{n-\frac{1}{2}}$. Again, the rate of change of strain and shear $\dot{\boldsymbol{\gamma}}_{n-\frac{1}{2}}$ required for the damping terms can be determined from the strain and shear vector $\boldsymbol{\gamma}_{n-\frac{1}{2}}$ by total differentiation with respect to time.

2.2.3 Internal Forces and Moments

Assuming linear elastic material behavior and viscous damping, the internal forces $\boldsymbol{f}_{n-\frac{1}{2}}$ assigned to the midpoints and internal moments $\boldsymbol{\ell}_n$ assigned to the vertices result in

$$\boldsymbol{f}_{n-\frac{1}{2}} = \boldsymbol{C}^\gamma \boldsymbol{\gamma}_{n-\frac{1}{2}} + 2 \cdot \boldsymbol{D}^\gamma \dot{\boldsymbol{\gamma}}_{n-\frac{1}{2}}, \tag{2.30}$$

$$\boldsymbol{\ell}_n = \boldsymbol{C}^\mathrm{K} \boldsymbol{k}_n + 2 \cdot \boldsymbol{D}^\mathrm{K} \dot{\boldsymbol{k}}_n. \tag{2.31}$$

with the stiffness matrix of elongation and shearing

$$\boldsymbol{C}^\gamma = \left[\begin{array}{c|ccc} 0 & & & \\ \hline & GA\kappa_1 & & \\ & & GA\kappa_2 & \\ & & & EA \end{array}\right] \tag{2.32}$$

and the stiffness matrix of bending and torsion

$$\boldsymbol{C}^K = \left[\begin{array}{c|ccc} 0 & & & \\ \hline & EI_1 & & \\ & & EI_2 & \\ & & & GJ \end{array}\right]. \tag{2.33}$$

The parameters of the stiffness matrices are the Young's modulus E and the shear modulus G, which are material parameters, as well as the second moments of area about the respective axes I_1 and I_2, the polar second moment of area J, the cross-sectional area A and the shear correction factors κ_1 and κ_2, which are geometric parameters. The shear correction factors κ_1 and κ_2 should not be confused with the vector of curvature and torsion $\boldsymbol{\kappa}_n$.

The damping matrix of elongation and shearing

$$\boldsymbol{D}^\gamma = \left[\begin{array}{c|ccc} 0 & & & \\ \hline & d_1^\gamma & & \\ & & d_2^\gamma & \\ & & & d_3^\gamma \end{array}\right] \tag{2.34}$$

and the damping matrix of bending and torsion

$$\boldsymbol{D}^K = \left[\begin{array}{c|ccc} 0 & & & \\ \hline & d_1^K & & \\ & & d_2^K & \\ & & & d_3^K \end{array}\right] \tag{2.35}$$

can usually only be determined experimentally. For simplicity, a diagonal matrix structure is assumed. The generalization to a non-diagonal structure is straightforward. However, for practical applications, damping matrices with a diagonal matrix structure are often preferred because this assumption simplifies the experimental determination of the damping parameters. Bending is often the most important deformation, and especially elongation is less important because soft robots are often very stiff in longitudinal direction. For this reason, in many applications it is sufficient to determine only the damping coefficients directly related to bending. The other damping coefficients should then be chosen sufficiently large to obtain good numerical properties [GrubeSeifried22b]. This work focuses on linear viscoelastic material models, as they are sufficient for many soft robotic applications. If needed, more complex material models can be inserted in a straightforward way. This is e.g. shown in [GrubeSeifried22b] for the Saint Venant–Kirchhoff material model and in [LinnEtAl13] for the integration of Kelvin–Voigt type viscous damping.

2.2.4 Equations of Motion

With the internal forces $\boldsymbol{f}_{n+\frac{1}{2}}$ and internal moments $\boldsymbol{\ell}_\nu$ derived in the equations (2.30) and (2.31), the balances of linear momentum and angular momentum can be established to derive the equations of motion in form of a differential-algebraic system of equations (DAE). With some transformations to take account of the peculiarities of quaternions, explained in [LangLinn09], this results in the balances of linear momentum

$$\rho A l_n \ddot{\boldsymbol{x}}_n = \left(\boldsymbol{p}_{n+\frac{1}{2}} \left(\boldsymbol{f}_{n+\frac{1}{2}} + \hat{\boldsymbol{f}}_{n+\frac{1}{2}}\right) \bar{\boldsymbol{p}}_{n+\frac{1}{2}} - \boldsymbol{p}_{n-\frac{1}{2}} \left(\boldsymbol{f}_{n-\frac{1}{2}} + \hat{\boldsymbol{f}}_{n-\frac{1}{2}}\right) \bar{\boldsymbol{p}}_{n-\frac{1}{2}}\right), \tag{2.36}$$

the balances of angular momentum

$$\frac{\rho L}{2} \boldsymbol{\mu}(\boldsymbol{p}_\nu) \ddot{\boldsymbol{p}}_\nu = \left(4\rho \dot{\boldsymbol{p}}_\nu \boldsymbol{I} \dot{\bar{\boldsymbol{p}}}_\nu \boldsymbol{p}_\nu + \Delta \boldsymbol{x}_\nu \boldsymbol{p}_\nu \boldsymbol{f}_\nu + \boldsymbol{p}_{\nu+1} \boldsymbol{\ell}_{\nu+\frac{1}{2}} - \boldsymbol{p}_{\nu-1} \boldsymbol{\ell}_{\nu-\frac{1}{2}} + \boldsymbol{p}_\nu \hat{\boldsymbol{\ell}}_\nu\right) - \boldsymbol{\lambda} \boldsymbol{p}, \tag{2.37}$$

and the quaternion constraints equations

$$0 = ||\boldsymbol{p}||^2 - 1. \tag{2.38}$$

The expression $\boldsymbol{\mu}(\boldsymbol{p}_\nu)$ is the quaternion mass matrix defined as

$$\boldsymbol{\mu}(\boldsymbol{p}_\nu) = 4 \boldsymbol{\mathcal{Q}}(\boldsymbol{p}_\nu) \boldsymbol{I} \boldsymbol{\mathcal{Q}}(\boldsymbol{p}_\nu)^\top \tag{2.39}$$

with the inertia matrix

$$\boldsymbol{I} = \begin{bmatrix} 0 & & & \\ & I_1 & & \\ & & I_2 & \\ & & & J \end{bmatrix} \tag{2.40}$$

and the quaternion matrix

$$\boldsymbol{\mathcal{Q}} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix}. \tag{2.41}$$

Here, I_1 and I_2 are the second moments of area about the x- and y-axis respectively and J is the polar moment of inertia with respect to the z-axis. In addition to the variables already discussed, the density of the beam ρ and the deformed length of the piece $\Delta \boldsymbol{x}_\nu = \boldsymbol{x}_{\nu+\frac{1}{2}} - \boldsymbol{x}_{\nu-\frac{1}{2}}$ occur.

Furthermore, the external forces $\hat{\boldsymbol{f}}_{n+\frac{1}{2}}$ and external moments $\hat{\boldsymbol{\ell}}_\nu$ occur in the equations of motion. Finally, the vector of Lagrange multipliers $\boldsymbol{\lambda}$ must be introduced because of the algebraic constraint equation (2.38), which enforces

the normality condition on the unit quaternions. Note that all products in the equations of motions are quaternion products.

By enforcing the quaternionic constraint on acceleration level only, the equations of motion can also be written in form of an ordinary differential system of equations (ODE) as follows

$$\ddot{\boldsymbol{x}}_n = \frac{1}{\rho A l_n}\left(\boldsymbol{p}_{n+\frac{1}{2}}\left(\boldsymbol{f}_{n+\frac{1}{2}} + \hat{\boldsymbol{f}}_{n+\frac{1}{2}}\right)\bar{\boldsymbol{p}}_{n+\frac{1}{2}} - \boldsymbol{p}_{n-\frac{1}{2}}\left(\boldsymbol{f}_{n-\frac{1}{2}} + \hat{\boldsymbol{f}}_{n-\frac{1}{2}}\right)\bar{\boldsymbol{p}}_{n-\frac{1}{2}}\right), \tag{2.42}$$

$$\begin{aligned}\ddot{\boldsymbol{p}}_\nu = \frac{2}{\rho L}\boldsymbol{\mu}(\boldsymbol{p}_\nu)^{\#}\left(4\rho\dot{\boldsymbol{p}}_\nu \boldsymbol{I}\dot{\bar{\boldsymbol{p}}}_\nu \boldsymbol{p}_\nu + \Delta\boldsymbol{x}_\nu \boldsymbol{p}_\nu \boldsymbol{f}_\nu + \boldsymbol{p}_{\nu+1}\boldsymbol{\ell}_{\nu+\frac{1}{2}} - \boldsymbol{p}_{\nu-1}\boldsymbol{\ell}_{\nu-\frac{1}{2}} + \boldsymbol{p}_\nu\hat{\boldsymbol{\ell}}_\nu\right)\\ - ||\dot{\boldsymbol{p}}_\nu||^2\,\boldsymbol{p}_\nu.\end{aligned} \tag{2.43}$$

Here the quaternionic constraint on acceleration level $\langle \boldsymbol{p}, \ddot{\boldsymbol{p}}\rangle = -\|\dot{\boldsymbol{p}}\|^2$ is included in equation (2.43). The expression $\boldsymbol{\mu}(\boldsymbol{p}_\nu)^{\#}$ is the pseudo-inverse of the inertia matrix defined by [LangEtAl11]. It is defined as

$$\boldsymbol{\mu}(\boldsymbol{p}_\nu)^{\#} = 4\boldsymbol{\mathcal{Q}}(\boldsymbol{p}_\nu)\boldsymbol{I}^{\#}\boldsymbol{\mathcal{Q}}(\boldsymbol{p}_\nu)^{\top} \tag{2.44}$$

with

$$\boldsymbol{I}^{\#} = \left[\begin{array}{c|ccc} 0 & & & \\ \hline & I_1^{-1} & & \\ & & I_2^{-1} & \\ & & & J^{-1} \end{array}\right] \tag{2.45}$$

and the quaternion matrix

$$\boldsymbol{\mathcal{Q}} = \left[\begin{array}{c|ccc} p_0 & -p_1 & -p_2 & -p_3 \\ \hline p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{array}\right]. \tag{2.46}$$

Due to the formulation of the quaternionic constraints on acceleration level only, quadratic numerical drift occurs when solving the ODE numerically. One approach to overcome this issue is the projection method. After each successful integration step, all quaternions and quaternion velocities are normalized to satisfy the constraint conditions. The normalized quantities are computed as

$$\boldsymbol{p}_i = \frac{\tilde{\boldsymbol{p}}_i}{||\tilde{\boldsymbol{p}}_i||}, \tag{2.47}$$

$$\dot{\boldsymbol{p}}_i = \dot{\tilde{\boldsymbol{p}}}_i - \tilde{\boldsymbol{p}}_i^{\top}\dot{\tilde{\boldsymbol{p}}}_i\tilde{\boldsymbol{p}}_i \tag{2.48}$$

[LangEtAl11]. The tilde is used to mark the quantities with drift. In this way, the numerical drift can be corrected after each iteration step. As an alternative, the Baumgarte stabilization, or a QR-decomposition of the equation of motion can be used. Finally, it is also possible to directly solve the DAE system from equations (2.36) to (2.38) with a suitable DAE solver.

In order to study the numerical drift and evaluate the effectiveness of the projection method, the oscillation of a weak cantilever beam around the equilibrium position under self-weight with the straight configuration as the initial condition and a discretization into N_{pieces} =5 pieces is considered as an example. The parameters of the beam are the same as for the beam with the weak parameter set introduced in Sec. 2.4.1 and described in more detail there. In Fig. 2.4, the maximum violation of the quaternionic constraints is plotted over time with and without the projection method. The constraint violation is shown for different values of the absolute integrator tolerances ε_{abs} and for the relative integrator tolerances ε_{rel}. As integrator `ode15s` in `Matlab` is used. It can be clearly seen that without projection the error of the quaternionic constraints grows constantly and is by far too large for practical applications. Reducing the integrator tolerances improves the results only slightly. With the projection method, the error can be reduced to $\Delta\,||\boldsymbol{p}_i|| = 3.88 \cdot 10^{-5}$ or $\Delta\,||\boldsymbol{p}_i|| = 1.35 \cdot 10^{-5}$ for the stricter integrator tolerances. Using the projection method, the violations of the constraints are sufficiently small that they can be tolerated for the simulation. This approach is used in the following.

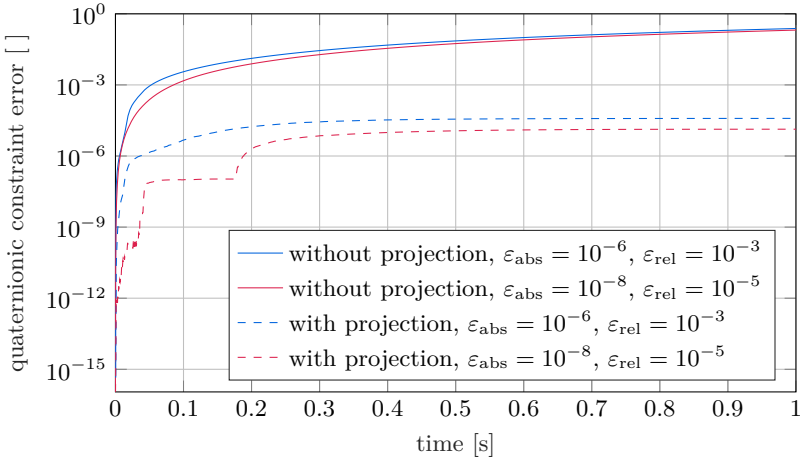


Figure 2.4: Quaternionic drift with and without projection method for different absolute and relative integrator tolerances.

2.2.5 Influence of the Scalar Characteristic Weighting Function ξ

Unlike translation interpolation and planar rotation interpolation, spatial rotation interpolation is not unique. In [LangEtAl11], based on [BauchauTrainelli03, BobenkoSuris99, CrisfieldJelenic99, Sander10, Shoemake85, SpillmanTeschner07], five different choices for the scalar characteristic weighting function are presented, which correspond to different interpolation schemes of spatial rotations. These are

$$\xi_1(\theta) = 1, \qquad \xi_2(\theta) = \sqrt{\frac{2}{1+\cos\theta}},$$
$$\xi_3(\theta) = \frac{1}{\cos\theta}, \qquad \xi_4(\theta) = \frac{2}{1+\cos\theta},$$
$$\xi_5(\theta) = \frac{\theta}{\sin\theta}.$$

Note that the domain of $\xi_3(\theta)$ is only $\{\theta \in \mathbb{R} : -\frac{\pi}{2} < \theta < \frac{\pi}{2}\}$, which is a limitation for extremely coarse discretizations, and that $\xi_5(\theta)$ has a removable singularity at $\theta = 0$. All the other scalar characteristic weighting functions have a domain of at least $\{\theta \in \mathbb{R} : -\pi < \theta < \pi\}$, which is sufficient for the vast majority of soft robotic applications. In Fig. 2.5, the proposed weighting functions are plotted over the angle θ. All functions are close to each other for small angles θ between successive pieces. Only for angles larger than $\frac{\pi}{18}$, which are $10°$, are there significant deviations in the values of the weighting functions. Thus, it can be expected that the choice of the weighting function is only relevant if the angle θ between successive pieces is sufficiently large. This is the case for a large total deflection and a coarse discretization.

To investigate the influence of the weighting function ξ, the deflection of a weak cantilever beam with a circular cross section and a total length of $L_{\text{total}} = 500\,\text{mm}$ is considered for a very coarse discretization of $N_{\text{pieces}} = 3$ and a slightly finer discretization of $N_{\text{pieces}} = 5$. The beam is aligned horizontally, clamped at one end, and loaded by its own weight. The parameters of the beam are the same as for the beam with the weak parameter set introduced in Sec. 2.4 and described in more detail there. It can be clearly seen that the influence of the discretization on the deflection is much larger than the influence of the choice of the weighting function ξ. For the coarse discretization with $N_{\text{pieces}} = 3$, the maximum difference in deflection between the different weighting functions is 5.82 mm, which is less than 1.4 % of the total deflection. For the finer discretization with $N_{\text{pieces}} = 5$ the difference is even smaller with only 2.3 mm, which is 0.6 % of the total deflection. The difference in deflection between the two discretizations $N_{\text{pieces}} = 3$ and $N_{\text{pieces}} = 5$ is much larger with a difference of 21.1 mm, which is 5.2 % of the total deflection. The influence of the discretization is examined in more detail in

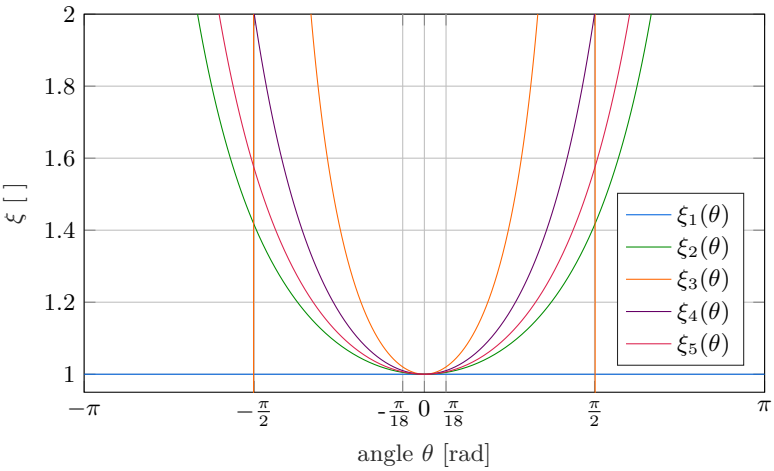


Figure 2.5: Possible choices for the scalar characteristic weighting functions ξ over the angle θ.

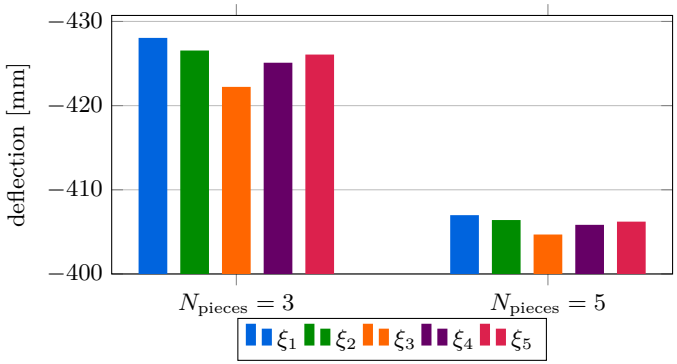


Figure 2.6: Influence of the choice of the scalar characteristic weighting functions ξ on the deflection in y-direction of a beam for different degrees of discretization N_{pieces}.

Sec. 2.4. Thus, it can be concluded that the choice of the weighting function ξ is of minor importance for typical soft robotic applications, since the influence is very small, even for a coarse discretization. Even if higher accuracy is required, typically a finer discretization must be chosen anyway. This in turn reduces the influence of the weighting function. In the following, the weighting function

$$\xi(\theta_n) = \xi_2(\theta_2) = \sqrt{\frac{2}{1+\cos\theta_2}} \tag{2.49}$$

is used, as described in Sec. 2.2.2. It has a domain of $\{\theta \in \mathbb{R} : -\pi < \theta < \pi\}$ and the function curve lies in the middle of the considered function curves.

2.3 Absolute Nodal Coordinate Formulation

The absolute nodal coordinate formulation (ANCF) introduced by [Shabana11] is a nonlinear FE approach from multibody dynamics for the modeling of large deformations. Depending on the specific formulation, ANCF beam models can model large bending, elongation torsional and shear deformations. On advantage of the ANCF formulation is that the mass matrix is constant. However, the calculation of the internal forces results in highly nonlinear terms. In the following, the description of the ANCF model follows [Shabana11].

2.3.1 Spatial Discretization

In Fig. 2.7, the discretization of a beam into N_{pieces} pieces using the ANCF model is shown. As an example, a two-noded three-dimensional beam element is shown. For each piece, the relationship between the current configuration and the undeformed reference configuration $\boldsymbol{X}$ is described by

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t). \tag{2.50}$$

In linear FEM this is a linear function, in the case of the nonlinear ANCF formulation the position vector $\boldsymbol{x}$ can be written as a polynomial of the form

$$x_i(\boldsymbol{X}, t) = a_0(t) + a_1(t)X_1 + a_2(t)X_2 + a_3(t)X_3 + \cdots + a_7(t)X_1^2 + \ldots \quad i = 1, 2, 3. \tag{2.51}$$

By applying separation of variables this can be written as

$$\boldsymbol{x}(\boldsymbol{X}, t) = \begin{bmatrix} \boldsymbol{p}_1(\boldsymbol{X}) & 0 & 0 \\ 0 & \boldsymbol{p}_2(\boldsymbol{X}) & 0 \\ 0 & 0 & \boldsymbol{p}_3(\boldsymbol{X}) \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{a}_1(t) \\ \boldsymbol{a}_2(t) \\ \boldsymbol{a}_3(t) \end{bmatrix} \tag{2.52}$$

with the polynomial basis $\boldsymbol{p}_i(\boldsymbol{X})$ and the time-varying coefficient vector $\boldsymbol{a}_i(t)$. Since the coefficients $\boldsymbol{a}(t)$ are difficult to interpret, they are replaced by generalized

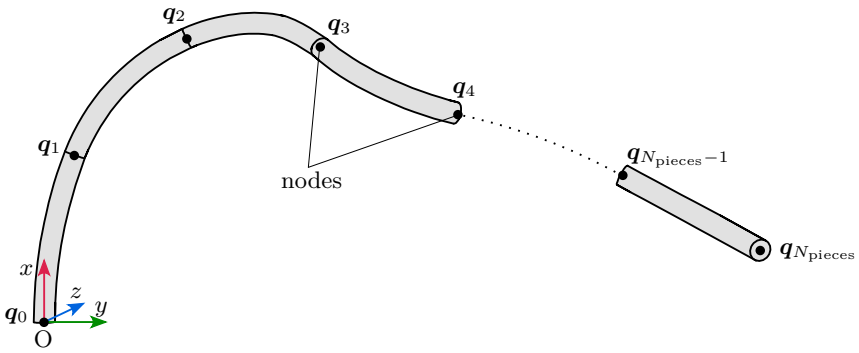


Figure 2.7: Discretization of a beam-shaped soft robot into N_{pieces} pieces for the ANCF model using a two-noded three-dimensional beam element.

nodal coordinates $\boldsymbol{q}$. These are the absolute position $\boldsymbol{x}$ of the node and the position gradients $\dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$. Equation (2.51) can then be written as

$$\boldsymbol{x}(\boldsymbol{X},t) = \boldsymbol{S}(\boldsymbol{X}) \cdot \left[\boldsymbol{r}^\top \quad \left(\frac{\partial \boldsymbol{x}}{\partial X_1}\right)^\top \quad \left(\frac{\partial \boldsymbol{x}}{\partial X_2}\right)^\top \quad \left(\frac{\partial \boldsymbol{x}}{\partial X_3}\right)^\top \right]^\top = \boldsymbol{S}(\boldsymbol{X})\boldsymbol{q}(t) \tag{2.53}$$

where $\boldsymbol{S}(\boldsymbol{X})$ is the shape function matrix that maps the generalized nodal coordinates $\boldsymbol{q}$ to the location vector of the current configuration. For good performance a suitable choice of the polynomial basis for the shape functions is crucial. Several possibilities are presented in [Shabana11]. In the following, the two-dimensional beam element from [Shabana11, Ch. 6.8] is used for planar deformations and the three-dimensional beam element from [Shabana11, Ch. 6.10] is used for 3D deformations. Both beam elements are briefly summarized below.

Two-Dimensional Beam Element

The two-dimensional beam element presented in [Shabana11, ch. 6.8] can represent planar bending and shear as well as strain along the longitudinal axis. Its shape function matrix $\boldsymbol{S}_{2\text{D}}$ is

$$\boldsymbol{S}_{2\text{D}} = \begin{bmatrix} s_1\boldsymbol{E} & s_2\boldsymbol{E} & s_3\boldsymbol{E} & s_4\boldsymbol{E} & s_5\boldsymbol{E} & s_6\boldsymbol{E} \end{bmatrix} \tag{2.54}$$

with the coefficients

$$\begin{aligned} s_1 &= 1 - 3\xi^2 + 2\xi^3, & s_2 &= l(\xi - 2\xi^2 + \xi^3), \\ s_3 &= l(\eta - \xi\eta), & s_4 &= 3\xi^2 - 2\xi^3, \\ s_5 &= l(-\xi^2 + \xi^3), & s_6 &= l\xi\eta. \end{aligned} \tag{2.55}$$

Here $\xi = \frac{X_1}{l}$, $\eta = \frac{X_2}{l}$ and l is the length of the element. A node described in this way has $f = 6$ degrees of freedom representing the displacement and the position gradients of the node, see equation (2.53). The two-dimensional beam element has two nodes which are each located in the middle of the two end faces of the beam. This two-dimensional ANCF element therefore has $f_{\text{beam2D}} = 12$ degrees of freedom.

Three-Dimensional Beam Element

The tree-dimensional beam element presented in [Shabana11, ch. 6.10] can represent bending and shear in two planes, as well as torsion and strain along an axis. Its shape function matrix $\boldsymbol{S}_{\text{3D}}$ is

$$\boldsymbol{S}_{\text{3D}} = \begin{bmatrix} s_1\boldsymbol{E} & s_2\boldsymbol{E} & s_3\boldsymbol{E} & s_4\boldsymbol{E} & s_5\boldsymbol{E} & s_6\boldsymbol{E} & s_7\boldsymbol{E} & s_8\boldsymbol{E} \end{bmatrix} \tag{2.56}$$

with the coefficients

$$\begin{aligned} s_1 &= 1 - 3\xi^2 + 2\xi^3, & s_2 &= l(\xi - 2\xi^2 + \xi^3), \\ s_3 &= l(\eta - \xi\eta), & s_4 &= l(\varsigma - \xi\varsigma), \\ s_5 &= 3\xi^2 - 2\xi^3, & s_6 &= l(-\xi^2 + \xi^3), \\ s_7 &= l\xi\eta, & s_8 &= l\xi\varsigma. \end{aligned} \tag{2.57}$$

Here $\xi = \frac{X_1}{l}$, $\eta = \frac{X_2}{l}$, $\varsigma = \frac{X_3}{l}$ and l is the length of the element.

A node described in this way has $f = 12$ degrees of freedom. The three-dimensional beam element has two nodes which are each located in the middle of the two end faces of the beam. This three-dimensional ANCF element therefore has $f_{\text{beam3D}} = 24$ degrees of freedom.

2.3.2 Equations of Motion

With the choice of the shape function, now the constant mass matrix $\boldsymbol{M}$ can be calculated. It results from the volume integral over the reference configuration Ω_0 with the density ρ as

$$\boldsymbol{M} = \int_{\Omega_0} \rho \boldsymbol{S}^\top \boldsymbol{S} \mathrm{d}V. \tag{2.58}$$

Similarly, external forces $\hat{\boldsymbol{f}}_i$ that act at the point $\boldsymbol{x}_i$ in the reference configuration and volume forces $\boldsymbol{b}$ can be described as

$$\boldsymbol{h}_{\text{p}} = \sum_i \boldsymbol{S}(\boldsymbol{x}_i)^\top \hat{\boldsymbol{f}}_i, \tag{2.59}$$

$$\boldsymbol{h}_{\text{b}} = \int_{\Omega_0} \boldsymbol{S}^\top \boldsymbol{b} \mathrm{d}V, \tag{2.60}$$

where $\boldsymbol{h}_\mathrm{p}$ and $\boldsymbol{h}_\mathrm{b}$ are the vectors of point- and volume forces, respectively. As there are no rotational coordinates, it is not possible to apply moments directly.

The elastic forces are determined from the internal stresses and strains. The Green-Lagrange strain tensor $\boldsymbol{\epsilon}$ is defined by

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\boldsymbol{J}^\top \boldsymbol{J} - \boldsymbol{I}\right). \tag{2.61}$$

Here $\boldsymbol{I}$ is the unit matrix and $\boldsymbol{J} = \dfrac{\partial(\boldsymbol{S}\boldsymbol{q})}{\partial \boldsymbol{X}}$ is the Jacobian matrix of the position vector $\boldsymbol{r} = \boldsymbol{S}\boldsymbol{q}$ of a point by the material coordinates $\boldsymbol{X}$. For a linear-elastic material law, the Piola-Kirchhoff stress tensor $\boldsymbol{\sigma}$ together with the elasticity tensor $\boldsymbol{E}$ results as

$$\boldsymbol{\sigma} = \boldsymbol{E} : \boldsymbol{\epsilon}. \tag{2.62}$$

The elastic forces can now be calculated from the volume integral

$$\boldsymbol{h}_\mathrm{e} = \int_{\Omega_0} \boldsymbol{\sigma} : \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{q}} \mathrm{d}V = \left(\int_{\Omega_0} \sum_{i,j} \sigma_{ij} \frac{\partial \epsilon_{ij}}{\partial \boldsymbol{q}} \mathrm{d}V \right)^\top. \tag{2.63}$$

The quantity σ_{ij} corresponds to the ij-th entry of the Piola-Kirchhoff stress tensor $\boldsymbol{\sigma}$, while ϵ_{ij} is the corresponding entry of the Green-Lagrangian strain tensor $\boldsymbol{\epsilon}$. In contrast to the mass matrix and the external loads, it is often impractical to evaluate the integral analytically as the integrand is strongly nonlinear. Numerical approaches to determining the integral are discussed in section 2.3.3.

The original formulation of [Shabana11] does not contain an approach to represent internal damping. In soft robotics, however, damping can usually not be neglected. Therefore, simple linear damping is added, as e.g. shown in [ČeponEtAl09]. Internal damping is formulated in the same way as elastic forces. First, the damping stress $\boldsymbol{\sigma}_\mathrm{d}$ is defined analogously to equation (2.62) as

$$\boldsymbol{\sigma}_\mathrm{d} = \boldsymbol{D} : \dot{\boldsymbol{\epsilon}} = \boldsymbol{D} : (\dot{\boldsymbol{J}}^\top \boldsymbol{J} + \boldsymbol{J}^\top \dot{\boldsymbol{J}}) \tag{2.64}$$

with the damping matrix $\boldsymbol{D}$. The change in strain $\dot{\boldsymbol{\epsilon}}$ is obtained by determining the total temporal differential of the strain $\boldsymbol{\epsilon}$ from equation (2.61). The internal damping force $\boldsymbol{h}_d$ follows from the damping stress analogously to equation (2.63) as

$$\boldsymbol{h}_\mathrm{d} = \int_{\Omega_0} \boldsymbol{\sigma}_\mathrm{d} : \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{q}} \mathrm{d}V = \left(\int_{\Omega_0} \sum_{i,j} \sigma_{\mathrm{d},ij} \frac{\partial \epsilon_{ij}}{\partial \boldsymbol{q}} \mathrm{d}V \right)^\top. \tag{2.65}$$

As for equation (2.63), equation (2.65) should be evaluated numerically.

By combining the equations (2.58), (2.59), (2.60) and (2.63) as well as (2.65), the equation of motion of an element is obtained as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{h}_\mathrm{p}(\boldsymbol{q}, t) + \boldsymbol{h}_\mathrm{b}(\boldsymbol{q}, t) - \boldsymbol{h}_\mathrm{e}(\boldsymbol{q}) - \boldsymbol{h}_\mathrm{d}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{2.66}$$

This can be easily assembled for a system consisting of several elements.

2.3.3 Numerical Evaluation of the Volume Integrals

The integrals (2.63) and (2.65) from Sec. 2.3.2 have to be evaluated numerically. This is done using the Gaussian quadrature method with Gauss-Legendre sampling points as described in [Shabana11, Ch. 6.5]. For an element with cylindrical shape, radius r and length ℓ, the volume integral of a function $\boldsymbol{f}(\boldsymbol{X})$ is given by

$$\int_{\Omega_0} \boldsymbol{f}(\boldsymbol{X})\mathrm{d}V = \int_0^\ell \int_{-r}^{r} \int_{-\sqrt{r^2-X_2^2}}^{\sqrt{r^2-X_2^2}} \boldsymbol{f}(X_1, X_2, X_3)\mathrm{d}X_3\mathrm{d}X_2\mathrm{d}X_1 \tag{2.67}$$

$$\approx \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} w_i w_j w_k \cdot \boldsymbol{f}(\tilde{r}_i, \tilde{r}_j, \tilde{r}_k) \cdot \frac{\ell r}{2} \cdot \sqrt{r^2 - \tilde{r}_j} \tag{2.68}$$

with

$$\tilde{r}_i = r_i \frac{\ell}{2} + \frac{\ell}{2}, \tag{2.69}$$

$$\tilde{r}_j = r_j r, \tag{2.70}$$

$$\tilde{r}_k = r_k \sqrt{r^2 - \tilde{r}_j}. \tag{2.71}$$

Here n_i is the order of the method for the corresponding coordinate direction with $i = 1, 2, 3$, r_m is the m-th grid point and w_m is the corresponding weight.

To select an integration order, the accuracy and computation time of integration orders from $1 \ldots 20$ are compared with each other. For this purpose, a cantilever beam discretized with one piece is simulated, which oscillates around the equilibrium position under its own weight. The parameters of the beam are the same as for the beam with the weak parameter set introduced in Sec. 2.4.1 and described in more detail there. In the following, the twentieth-order method is used as reference solution and for the simulation to determine the elastic forces. The results of the simulation are then used to calculate the results of the remaining orders for each time step.

As an error measure, the relative root mean squared error $e_{n,\mathrm{rms}}$

$$e_{n,\mathrm{rms}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{||\boldsymbol{h}_{\mathrm{e},n}||_i}{||\boldsymbol{h}_{\mathrm{e},20}||_i} - 1\right)^2} \tag{2.72}$$

of the elastic forces $\boldsymbol{h}_{\mathrm{e,n}}$ introduced in equation (2.63) with respect to the reference solution $||\boldsymbol{h}_{\mathrm{e},20}||$ with integration order 20 is used. The error $e_{n,\mathrm{rms}}$ is plotted in Fig. 2.8 over the Gauss integration order. Additionally, in Fig. 2.9, the average calculation time for determining the integral value is shown for different integration orders. It can be clearly seen that the error decreases with increasing order while the computational effort increases cubically. In the following, the eighth-order method is used. It is the first method to have an error $e_{n,\mathrm{rms}} < 10^{-3}$. The time required to evaluate the integral of the elastic forces $\boldsymbol{h}_{\mathrm{e}}$ in one time step is 0.23 ms, which is acceptable for this work.

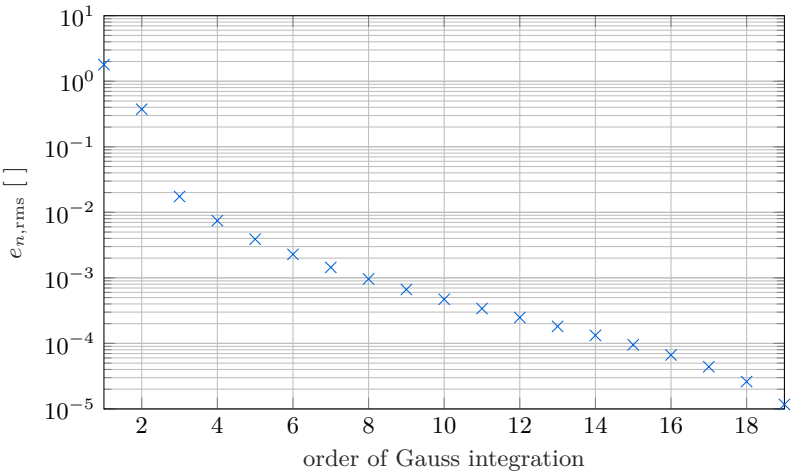


Figure 2.8: Relative mean square error $e_{n,\mathrm{rms}}$ over the Gauss integration order.

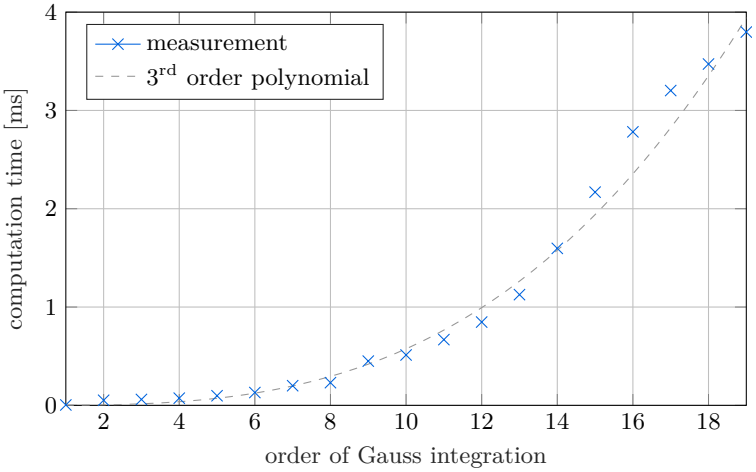


Figure 2.9: Evaluation time of the integral of the elastic forces $\boldsymbol{h}_\mathrm{e}$ over the Gauss integration order.

2.4 Model Accuracy

In the following, first a test system used to evaluate the performance of the beam models presented in Secs. 2.1 to 2.3 is presented. Then, three different simulation scenarios are considered to evaluate the kinematic and dynamic accuracy and the influence of the discretization for the three different beam models.

2.4.1 Test System for Evaluation of Beam Models

As a test system, a horizontal cantilever beam shown in Fig. 2.10 with density ρ, length L_tot and a constant circular cross section with radius r is considered. The circular cross section gives a shear correction factor of $\kappa = 3/4$ [Timoshenko40]. The shear correction factor κ is necessary for the CR model. The Young's modulus is chosen to be similar to that of Dragon Skin [Smooth-On24], a silicone material widely used in soft robotics. Silicone is nearly incompressible, resulting in a Poisson's ratio of $\nu \approx 0.49$. However, since the ANCF model runs into shear locking for large values of ν, a value of $\nu = 0$ is also considered for the ANCF model. For the simulations, two different sets of parameters listed in Tab. 2.1 are considered, which differ in their length and radius and thus also in their bending stiffness EI_{xx}. Both sets together represent a wide range of different soft robots.

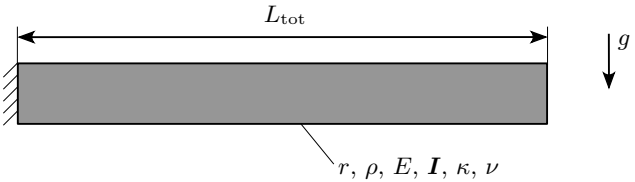


Figure 2.10: Test system used to evaluate the performance of the beam models in simulation.

The damping of the beam varies depending on the simulation scenario. For the determination of the equilibrium position in Secs. 2.4.2 and 2.4.4, critical damping is assumed because critical damping leads to a fast convergence and the dynamics of the system are not important there. For the determination of the eigenfrequencies of the system in Secs. 2.4.3 and 2.4.4, the system is assumed to be undamped. This allows to use the analytical solution of the Euler-Bernouli vibration differential equation as a reference solution. Finally, in Sec. 2.5 weak stiffness-proportional damping is assumed, as this is realistic for many soft robotic applications. A value of $\boldsymbol{D} = 25 \cdot 10^{-4}\,\mathrm{s} \cdot \boldsymbol{K}$, where $\boldsymbol{D}$ is the damping matrix and $\boldsymbol{K}$ is the stiffness matrix is chosen.

Note that the symbolic derivation of the 3D PCC model is computationally very intensive for fine discretizations, which limits the maximum number of pieces. In this work, a maximum number of 5 pieces is used. For the 2D version of the PCC model, this is less of an issue. Therefore, the 2D versions of the PCC and ANCF model are used for the purely planar simulation scenarios. Additionally, these models are computationally more efficient than the 3D versions. For the formulation of the CR model used in this work, there is no 2D version, so the 3D version is always used. The 3D version of all models is used for the spatial simulation scenarios. For the spatial simulation scenarios, the finest discretization considered is 5 pieces. For the 2D simulation scenarios a discretization in up to 20 pieces is used.

Table 2.1: Material and geometric parameters for both parameter sets.

symbol	description	weak parameter set	stiff parameter set
r	radius	$30\,\mathrm{mm}$	$40\,\mathrm{mm}$
ρ	density	$1070\,\mathrm{kg/m^3}$	$1070\,\mathrm{kg/m^3}$
E	Young's modulus	$4.12 \cdot 10^5\,\mathrm{Pa}$	$4.12 \cdot 10^5\,\mathrm{Pa}$
L_{tot}	length	$500\,\mathrm{mm}$	$200\,\mathrm{mm}$
EI_{xx}	bending stiffness	$2.62 \cdot 10^{-1}\,\mathrm{Nm^2}$	$8.28 \cdot 10^{-1}\,\mathrm{Nm^2}$
κ	shear correction factor	$3/4$	$3/4$
ν	Poisson's ratio	0.49 / 0 (ANCF)	0.49 / 0 (ANCF)

2.4.2 Deflection under Self-Weight

In the first simulation scenario, the static deflection under self-weight is investigated. Since this is a purely planar load case, the 2D versions of the ANCF and PCC models and the 3D CR model are used. The numerical simulation is started from the undeformed initial position of the beam and is stopped when the oscillations of the beam have decayed due to damping. The termination condition is defined as $||\dot{\boldsymbol{y}}|| < 1 \cdot 10^{-5}\,\mathrm{m/s}$ and $||\ddot{\boldsymbol{y}}|| < 1 \cdot 10^{-5}\,\mathrm{m/s^2}$. In addition to the three dynamic beam models, the analytic Euler-Bernoulli and Timoshenko beam models are considered as reference solutions for the stiff parameter set. The deflection curve $w(x)$ is given by

$$w_{\mathrm{EB}}(x) = \frac{q}{EI_{xx}} \left(\frac{1}{24} x^4 - \frac{1}{6} L_{\mathrm{tot}} x^3 + \frac{1}{4} L_{\mathrm{tot}}^2 x^2 \right) \tag{2.73}$$

for the Euler-Bernoulli beam and

$$w_{\mathrm{T}}(x) = \underbrace{\frac{q}{EI_{xx}} \left(\frac{1}{24} x^4 - \frac{1}{6} L_{\mathrm{tot}} x^3 + \frac{1}{4} L_{\mathrm{tot}}^2 x^2 \right)}_{\text{bending component}} + \underbrace{\frac{q}{\kappa GA} \left(-\frac{1}{2} x^2 + L_{\mathrm{tot}} x \right)}_{\text{shear component}} \tag{2.74}$$

for the Timoshenko beam [GrossEtAl12]. Here $q = -A\rho g$ is the line load due to gravity with the circular cross section area $A = \pi r^2$ and $x \in [0, L_{\mathrm{tot}}]$ is the beam coordinate. There is no analytical solution for the weak parameter set since the resulting deflection is too large.

In Fig. 2.11 the maximum deflection is shown for different discretizations. For the stiff configuration, it can be seen that for an increasingly finer discretization, the PCC model converges to the analytic Euler-Bernoulli solution, and both the Cosserat rod model and the ANCF model with $\nu = 0$ converge to the Timoshenko solution. The convergence to the two different solutions can be explained by the fact that the PCC model and the Euler-Bernoulli model do not include shear

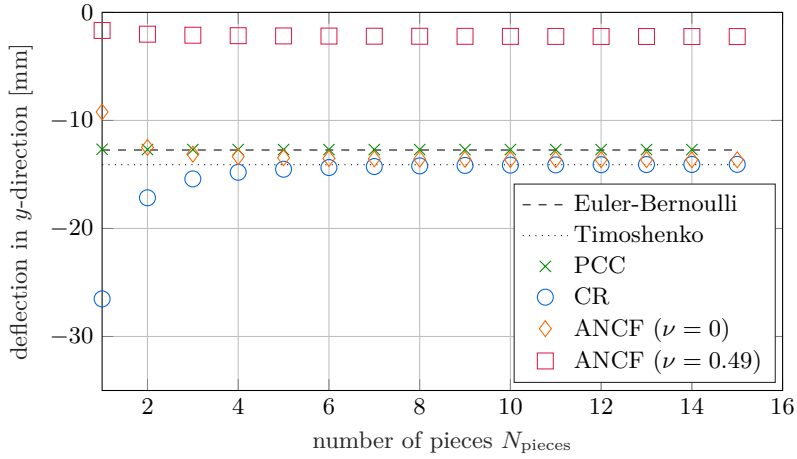
deformation, while the Cosserat model, the ANCF model, and the Timoshenko model do.

The ANCF model with a Poisson's ratio of $\nu = 0.49$ shows a much stiffer solution than the other models. This can be explained by element shear locking, which is a typical problem of many FEM formulations for modeling materials that are nearly incompressible [Bathe14, Ch. 4.5]. Typical ways to overcome the element shear locking are the use of higher order Ansatz functions, which comes at the cost of higher computational effort, or the use of mixed or reduced integration [Bathe14, Ch. 5.4]. Alternatively, Poisson's ratio reduction is possible if its influence is sufficiently small. This is typically the case for most bending scenarios considered in soft robotics and is therefore used in the following. For the CR model there is no locking, the solution for $\nu = 0$ and $\nu = 0.49$ are almost identical. For clarity, only the results for $\nu = 0$ are shown here and in the following. Also, the PCC model is not affected by element shear locking because shear is not included in the model.

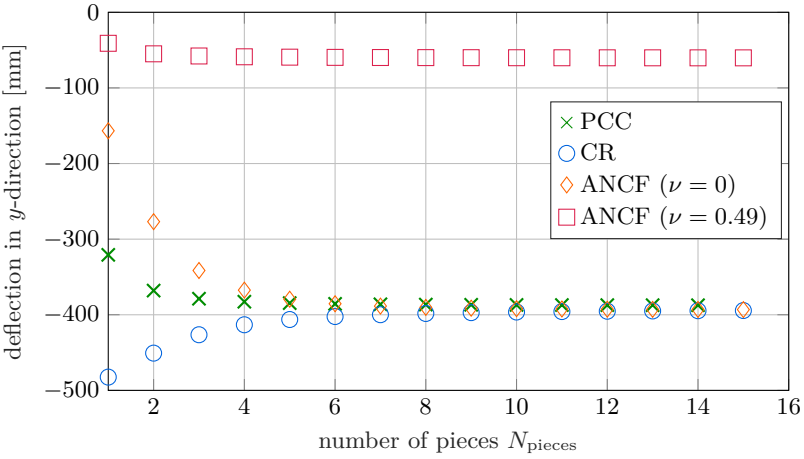
For the weak parameter set, a convergence of the ANCF ($\nu = 0$), the PCC and the CR model to approximately the same deflection can be observed for an increasingly finer discretization.

On the weak parameter set, the convergence of the PCC model is fastest. A discretization into only three pieces is required to achieve a deviation of less than 5 % from the converged value. The CR and ANCF ($\nu = 0$) models require a 5-piece discretization to achieve the same accuracy. For practical soft robotic applications, this is a very good accuracy. Most likely the simulation error due to unmodeled effects such as manufacturing inaccuracies, wear, or complex nonlinear material behavior is much larger. On the stiff parameter set, the convergence behavior is qualitatively comparable. It should be emphasized that with the PCC model, a deviation from the Bernoulli model of about 0.5 % can already be achieved with a single-piece discretization. Note that for both parameter sets, the CR model underestimates the stiffness of the beam, while the ANCF and PCC models overestimate it.

In Fig. 2.12 the deflection curve of the weak parameter set is shown for the ANCF ($\nu = 0$), the PCC and the CR model using a discretization with five pieces in each case. As for the maximum deflection, also the deflection curve is very similar for all three methods. The figure also shows how curvatures are represented by the different methods. While the PCC forms circular arcs, the shape of a segment in the ANCF is described by a cubic function. The segments of the discretized CR model are always straight. The curvature is represented here by rotating two adjacent segments relative to each other.



a) Stiff parameter set.



b) Weak parameter set.

Figure 2.11: Equilibrium position of the free end for different models and discretizations.

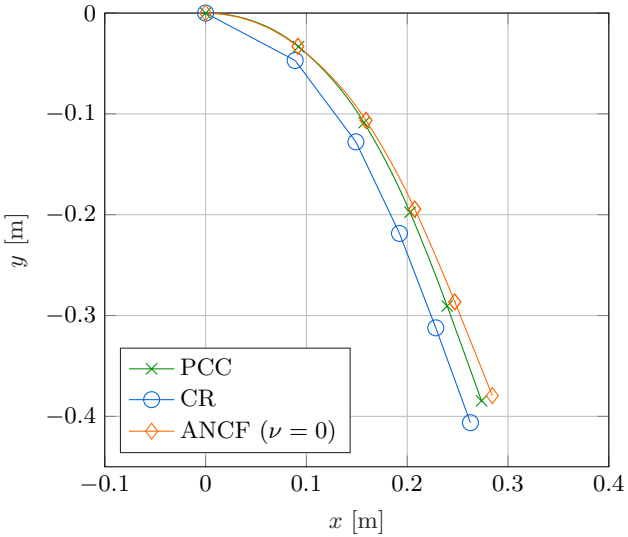


Figure 2.12: Deflection curve of the weak parameter set under self-weight for the five-piece discretization for the different simulation models.

2.4.3 First Bending Eigenfrequency

In the second simulation scenario, the dynamic properties of the different models and discretizations are examined. Therefore, the first bending eigenfrequency is determined in a simulation and compared to the analytic Euler-Bernoulli solution. The analytical Euler-Bernoulli solution is also used to determine the initial conditions for the numerical simulation. Again, two different cantilever beams with the parameters listed in Tab. 2.1 are considered. The analytic first eigenfrequency of a Euler-Bernoulli beam is

$$\omega_1 = \lambda^2 \cdot \sqrt{\frac{EI_{xx}}{\rho A}} \tag{2.75}$$

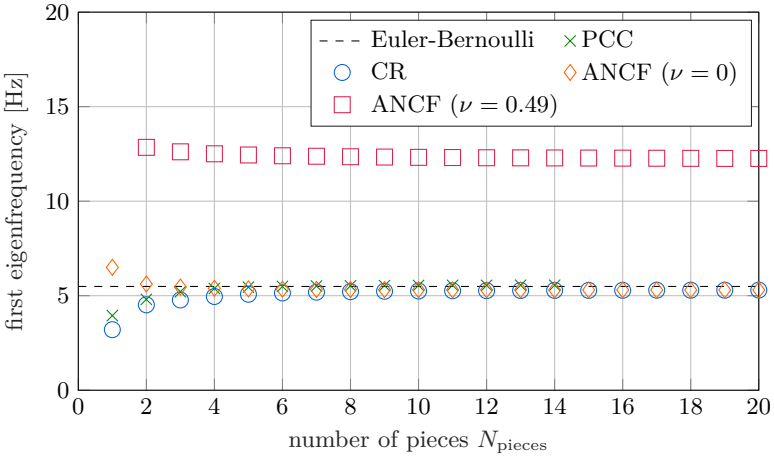
and the corresponding first normal mode is

$$w(t,x) = \frac{1}{2} q_1 \left(\frac{\cosh(\lambda x) - \cos(\lambda x)}{\cosh(\lambda L) - \cos(\lambda L)} - \frac{\sinh(\lambda x) - \sin(\lambda x)}{\sinh(\lambda L) - \sin(\lambda L)} \right) \cdot \cos(\omega_1 t) \tag{2.76}$$

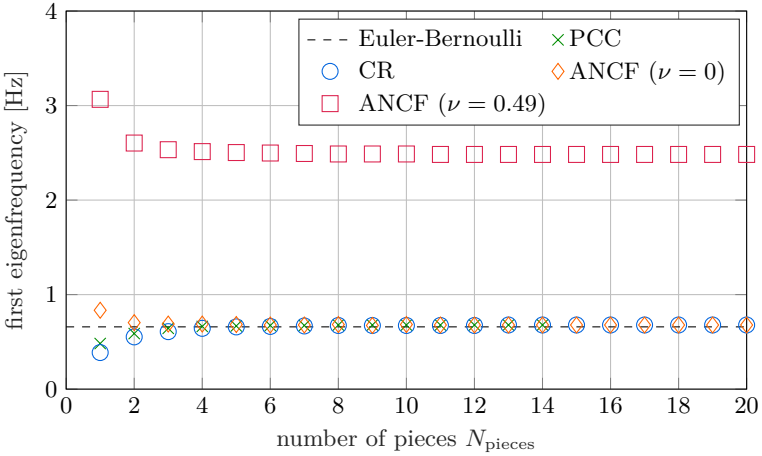
with λ as the eigenvalue of the beam differential equation and q_1 as a scaling factor [GaschEtAl21]. For the first eigenfrequency $\lambda L = 1.875$ holds. In the following, the scaling factor $q_1 = \frac{25}{2000} L$ is chosen, which corresponds to a small deflection of the beam of 2.5 % of the total length of the beam.

For the numerical determination of the eigenfrequency, the simulation is started with the deflection in the form of the first eigenfrequency given by $w(0,x)$ from equation (2.76) and is run for two full periods of the oscillation. Gravity is neglected for simplicity. The fast Fourier transformation (FFT) is then used to determine the first eigenfrequency, which is the dominant frequency in the simulation data.

In Fig. 2.13 the determined eigenfrequencies for the different models and both parameter sets are shown. For the ANCF model with a Poisson's ratio of $\nu = 0.49$, locking can be observed as for the deflection under self-weight in Sec. 2.4.2 . In all other simulation cases considered good convergence can be observed. For the finest discretization investigated for all models with $N_{\text{pieces}} = 14$, the deviation between the individual numerical methods is less than 0.5 % and the deviation from the analytical solution is less than 4 % for all models and both parameter sets. For all models except the CR model in the stiff configuration, this accuracy is already achieved with a discretization of only $N_{\text{pieces}} = 5$ pieces. Note that the analytical solution is not necessarily more accurate, since, for example, elongation and shear deformation are not included in the Euler-Bernoulli model but are included in some of the numerical models.



a) Stiff parameter set.



b) Weak parameter set.

Figure 2.13: Eigenfrequency for different discretizations and simulation models.

2.4.4 Torsion

The third simulation scenario examines torsion of a cantilever beam. First, the torsion under a constant external moment and then the first torsional eigenfrequency is examined. Note that the 3D version of all models is used here, since torsion is not included in the 2D models. For the sake of clarity, only the weak

parameter set is considered here, and gravity is neglected. The results for the stiff parameter set are qualitatively comparable.

For the examination of the torsion under a constant external moment, a moment of $M_{z,\text{ext}} = 1 \cdot 10^{-1}\,\text{Nm}$ is applied at the free end of the cantilever beam. The analytical solution for the torsion at the free end of the beam is given by

$$\vartheta_{\max} = \frac{M_{z,\text{ext}} L}{G I_{\text{P}}} = 19.08\,\text{mrad} \tag{2.77}$$

with the polar second moment of area I_{P} for a circular cross section

$$I_{\text{P}} = \frac{\pi r^4}{2}, \tag{2.78}$$

see e.g. [GrossEtAl12, Ch. 5]. The calculated torsion for all methods and different discretizations is shown in Fig. 2.14. The ANCF and the PCC method show a very good agreement with the analytical solution even for a discretization with just one piece. The CR method takes much more elements to converge. This can be explained by the staggered grid discretization, which leads to the fact that the external moment $M_{z,\text{ext}}$ cannot be applied at the end of the beam, but only at the center of the last piece. As a result, the torsional stiffness of the beam is systematically overestimated by a factor of $\frac{1}{2N_{\text{pieces}}}$, where N_{pieces} is the number of pieces used for discretization.

To investigate the dynamic torsional properties, the beam is twisted by an angle of $\Theta_{\text{tot}} = 5°$ and released from this initial position. Analogous to the procedure

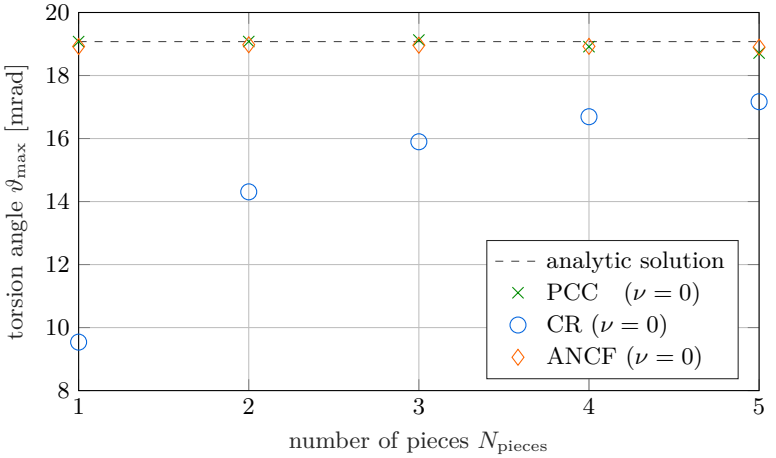


Figure 2.14: Torsion angle under constant external moment for different discretizations and simulation models.

in Sec. 2.4.3, two vibration periods are considered and the oscillation frequency is determined using an FFT. The simulation results for the different models, different values for the Poisson's ratio ν and different discretizations are shown in Fig. 2.15. It can be seen that both the ANCF and PCC models converge. There is good agreement even with a single-piece discretization. The resulting frequency depends on ν because the shear modulus G depends on ν. The CR model, however, shows a much stiffer behavior and does not converge to the value of the other two methods. This can be explained by numerical issues of the implementation of the CR model.

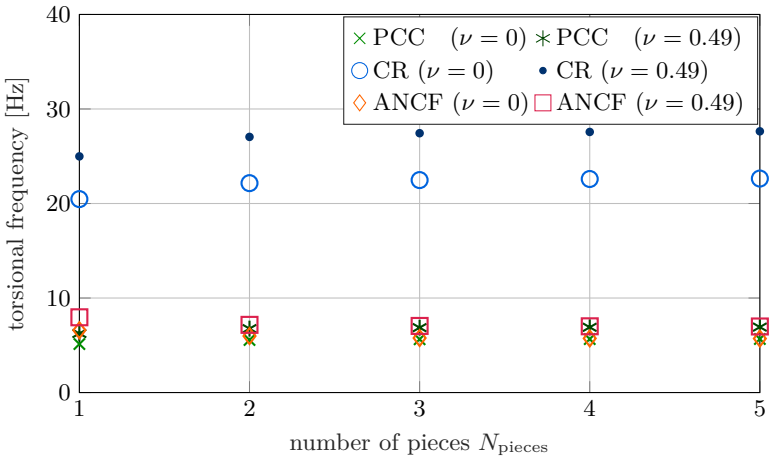


Figure 2.15: Eigenfrequency of the torsional oscillation for different discretizations and simulation models.

2.5 Computational Efficiency

Finally, the computational efficiency of the three beam models is compared. In a first step, the computation time for the CR model and the 2D versions of the ANCF and PCC models is examined in detail for different discretizations. As an example, the deflection under self-weight for the weak configuration as described in Sec. 2.4.2 is considered. The resulting oscillations are simulated over 5 s. The integrator tolerances are set to the default values which are $e_{\mathrm{AbsTol}} = 1 \cdot 10^{-6}$ for the absolute tolerance and $e_{\mathrm{RelTol}} = 1 \cdot 10^{-3}$ for the relative tolerance. The stiff `ode15s` solver is used for the ANCF and CR model, and the explicit `ode45` solver is used for the PCC model, as these are the MATLAB solvers that performed best for this particular problem. All calculations are performed on an Intel Core i7 - 6700K processor.

The computation time is influenced by the evaluation time of the ODE function, the number of required function evaluations, and the overhead of the integrator. For the simulation scenario considered, the overhead of the integrator is about $2.34 \cdot 10^{-6}$ s per function evaluation for the ode45 integrator and $1.26 \cdot 10^{-5}$ s per function evaluation for the ode45 integrator. This is one to two orders of magnitude less than the time for the function evaluation itself and is therefore not considered further in the following.

2.5.1 Evaluation Time of the ODE Functions

To investigate the computation time of the ODE functions, the ODE functions are executed 5000 times with random values for the state vector. For the CR model, the random values are generated such that the quaternion conditions are satisfied. The ODE functions are considered in a compiled (MEX) and an uncompiled version. The mean execution times are shown in Fig. 2.16.

The evaluation time of the ANCF model is by far the largest for all methods considered and increases linearly with an increasingly finer discretization. The main reason for the high computation time is the evaluation of the integrals (2.63) and (2.65). The CR model has the lowest evaluation time per function call, which also increases linearly with finer discretization. Except for the one-piece discretization, the compiled version of the ODE function is much faster than the uncompiled one. The longer evaluation time for the compiled version of the ODE function with a single-piece discretization can be explained by the overhead for the function call of the compiled function. The PCC model has a slightly longer evaluation time than the CR model for a discretization into a few pieces. However, as the discretization becomes finer, the evaluation time increases exponentially, which leads to a strong increase of the computation time. This can be explained by the complex structure of the equations of motion, which

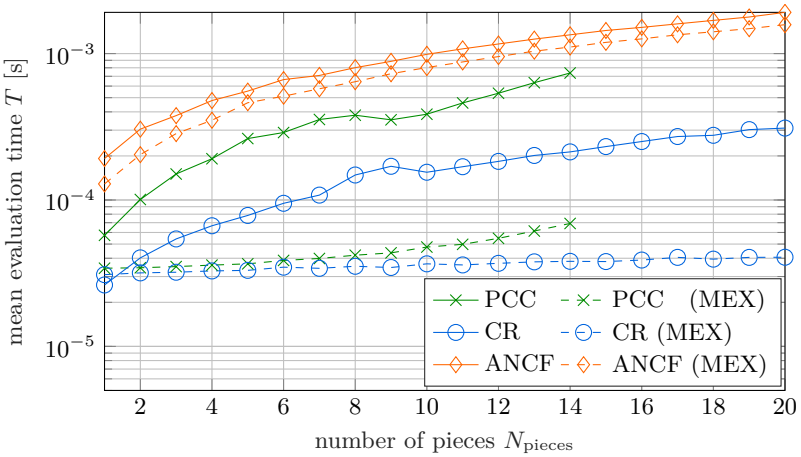


Figure 2.16: Mean evaluation time of the ODE function in uncompiled and compiled (MEX) version over the number of pieces N_{pieces} for different simulation models.

leads to computationally intensive ODE functions for fine discretizations. The PCC model benefits most from the compilation of the ODE functions. For fine discretizations, the compiled version is more than ten times faster.

2.5.2 Number of ODE Function Calls

The second parameter that determines the total computation time is the number of evaluations of the ODE function required. The results are shown in Fig. 2.17. The CR model requires the most function evaluations. This value increases slightly with a finer discretization. The reason for the high number of function calls is the stiffness of the equations of motion, which leads to the need for an implicit integrator such as `ode15s`, which requires many function calls to numerically determine the Jacobian matrix. Note that the `ode15s` integrator does not compute the Jacobian in every iteration, but reuses the previously computed Jacobian whenever possible [ShampineReichelt97]. The behavior of the ANCF model is similar to the CR model, but slightly fewer function evaluations are required. This can be explained by the fact that the Jacobian changes more slowly than for the CR model and thus requires fewer calculations of the Jacobian, which results in fewer function evaluations. The PCC model requires by far the least number of function evaluations for a discretization with few pieces. This can be explained by less stiff equations of motion and the use of an explicit solver, which does not require the computation of a Jacobian. However, the number of required function evaluations increases rapidly with an increasingly finer discretization.

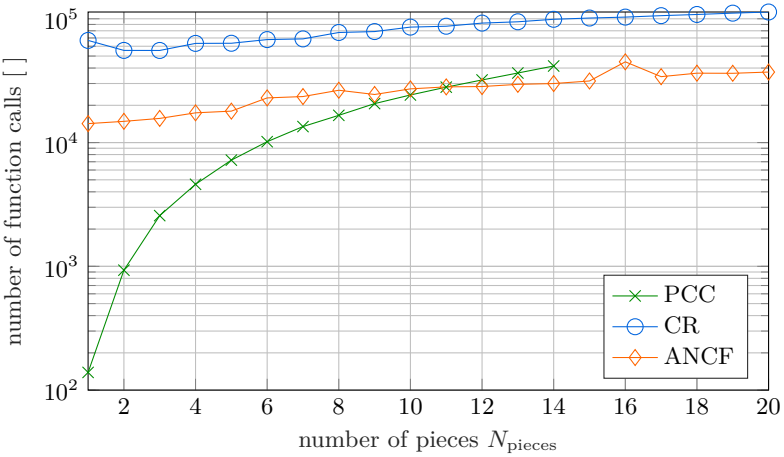


Figure 2.17: Number of the ODE function evaluations over the number of pieces N_{pieces} for different simulation models.

For more than $N_{\text{pieces}} = 11$ pieces, even more function evaluations are needed than for the ANCF model. This can be explained by the complex equations of motion for fine discretizations, which require the solver to use smaller time steps.

2.5.3 Theoretical Total Computation Time

The theoretical total computation time is here considered as the product of the time per ODE function evaluation and the number of function evaluations. Thus, it does not include the effects of the solver overhead. The result is shown in Fig. 2.18. It can be seen that the total computation time for the PCC and the CR model is very similar. All investigated discretizations of the compiled versions of these models simulate faster than real time. The PCC model is slightly faster. However, as the discretization becomes finer, the computation time increases faster than for the CR model. The uncompiled versions take a much longer computation time. The ANCF model has the highest computation time of the three models for both the compiled and the uncompiled versions. For the ANCF model, the difference between the compiled and uncompiled versions is small. The compiled ANCF model is not real-time capable for more than three pieces. Note that the required computation time for the models depends on the simulation scenario. Especially for the ANCF model, it is known that it performs best for very slender systems with low stiffness and high density [Shabana11, Ch. 6], which leads to low eigenfrequencies. However, these ideal conditions are usually not met for real soft robotic systems, as also seen here.

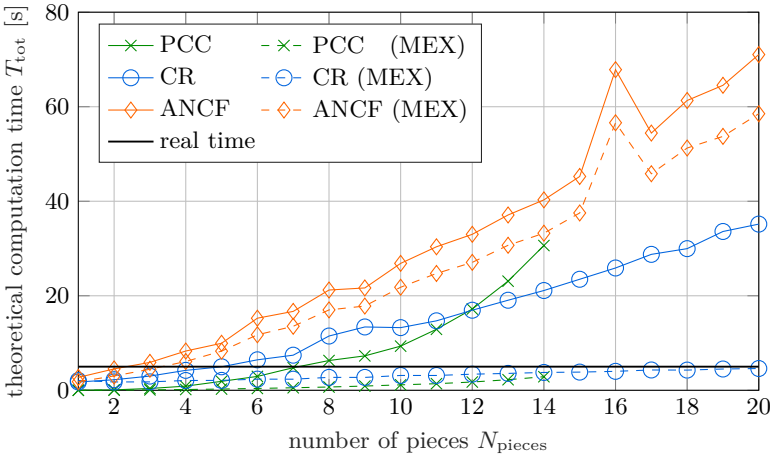


Figure 2.18: Theoretical total computation time with uncompiled and compiled (MEX) ODE function over the number of pieces N_{pieces} for different simulation models.

2.5.4 Computation Time for a 3D Trajectory

Finally, the real computational performance including the solver overhead is examined for a 3D spiral trajectory, see Fig. 2.22. Therefore, the simulation is started in straight configuration. The beam is loaded with a force $\boldsymbol{F}_{\text{tip}}$ given by

$$\boldsymbol{F}_{\text{tip}} = \begin{bmatrix} 20\,\text{N}\sin(\omega t)\frac{t}{T_{\max}} \\ 20\,\text{N}\cos(\omega t)\frac{t}{T_{\max}} \\ 0 \end{bmatrix} \tag{2.79}$$

acting at the tip of the beam with the angular velocity $\omega = 10\,\text{rad/s}$ and the total simulation time is $T_{\max} = 3\,\text{s}$.

Since no analytical solution is available, the mean of the simulation results calculated by the three simulation models with the finest discretization of $N_{\text{PCC}} = 5$ for the PCC model and $N_{\text{CR}} = N_{\text{ANCF}} = 20$ for the CR and the ANCF model is chosen as the reference solution. Note that there is no finer discretization available for the PCC model because the derivation of the equations of motion is computationally too expensive. As error measures the differences e_{mean} and $e_{\max}$ are defined as

$$e_{\text{mean}} = \text{mean}(e(t)), \tag{2.80}$$

$$e_{\max} = \max(e(t)) \tag{2.81}$$

with

$$e(t) = ||\boldsymbol{r}_{\text{tip}}(t) - \boldsymbol{r}_{\text{tip,ref}}(t)||_2 \,. \tag{2.82}$$

Here $\boldsymbol{r}_{\text{tip}}(t)$ is the simulated tip position and $\boldsymbol{r}_{\text{tip,ref}}(t)$ is the reference solution. For all three simulation models the mean difference e_{mean} to the reference solution is smaller than 0.9 mm and the maximum difference e_{max} to the reference solution is smaller than 1.6 mm. For practical soft robotic applications, this is a very good accuracy. Most likely the simulation error due to unmodeled effects such as manufacturing inaccuracies, wear, or complex nonlinear material behavior is much larger.

In Fig. 2.19 the computation time is shown for all three simulation methods and for different discretizations. Comparable to the results for the 2D case, the ANCF model is by far the slowest model and, except for a discretization with only 1 piece, always slower than real time. Both, the CR and the PCC model always simulate faster than real time. The PCC model is by far the fastest model for coarse discretizations, but the computation time increases strongly for finer discretizations. The CR model has a higher computation time than the PCC model for coarse discretizations, but the computation time increases much slower. For a discretization of $N_{\text{pieces}} = 5$, the computation time for both models is almost identical. Note again that this is the finest discretization available for the 3D PCC model.

In Fig. 2.20 the mean difference to the reference solution is shown for all three simulation models and for different discretizations. It can be seen that all models

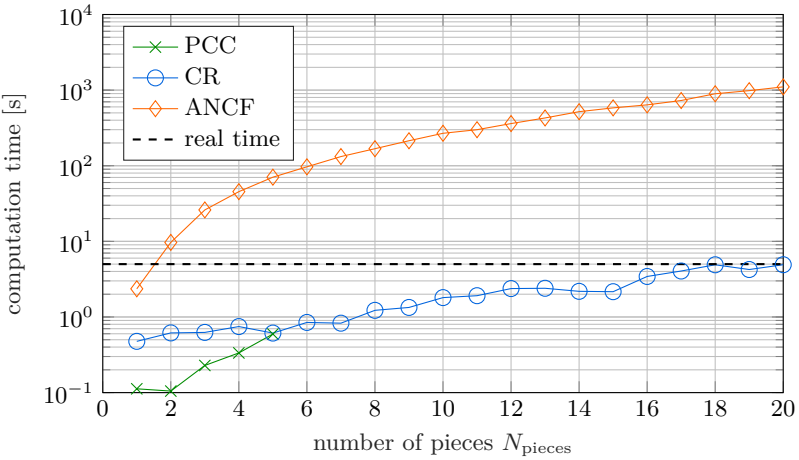


Figure 2.19: Computation time of the spiral trajectory for different beam models and discretizations.

converge approximately to the reference solution. The PCC and the ANCF model converge much faster than the CR model. Note that for the ANCF model the smallest difference to the reference solution is achieved for $N_{\text{pieces}} = 4$ pieces, for finer discretizations the difference slightly increases. This can be explained by the fact that the ANCF model converges to a slightly different solution than the other two models.

In the following, the discretization is chosen such that the mean difference to the reference solution is below 2 mm and the maximum difference to the reference solution is below 5 mm. This results in a discretization into $N_{\text{pieces}} = 2$ pieces for the PCC model and the ANCF model and a discretization into $N_{\text{pieces}} = 7$ pieces for the CR model. In Fig. 2.21 the deflection in x-direction is plotted over time for all three models. The behavior in y-direction is qualitatively identical and therefore omitted here. All three models show a very good agreement, however, compared to the reference solution the amplitude is slightly underestimated. This can also be seen in the $x - y$-plot shown in Fig. 2.22.

For real-time applications, it is usually not the total computation time that is important, but rather the computation time for each time step. Therefore, low-order solvers with a fixed step size, such as the implicit Euler method, are often used. In Tab. 2.2 the computation time of the implicit Euler method is given for the spiral trajectory and all three simulation models. Different step sizes are considered, so that a mean difference to the reference solution of $10 \cdot 10^{-2}$ mm, $10 \cdot 10^{-3}$ mm and $10 \cdot 10^{-3}$ mm is achieved. For reference, the computation times of the `ode15s` and `ode45` solvers, which both use an adaptive step size, are also

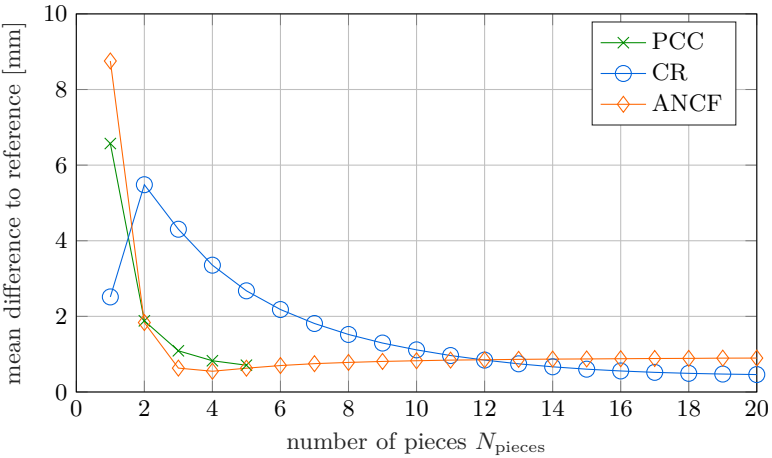


Figure 2.20: Mean difference to the reference solution of different simulation models for the spiral trajectory using different discretizations.

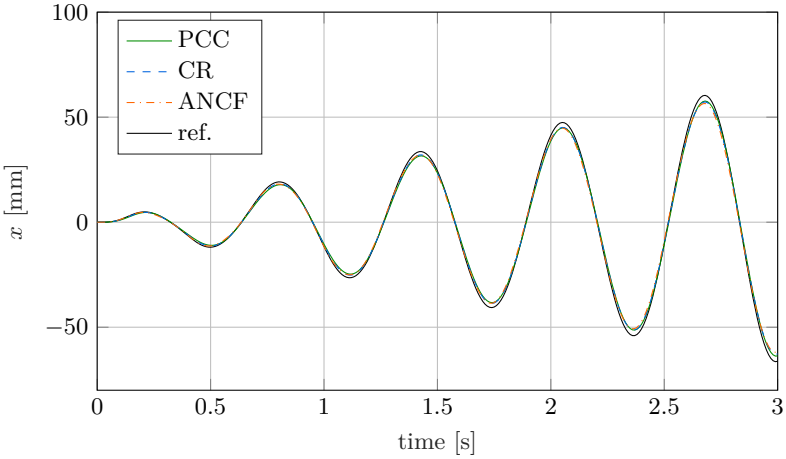


Figure 2.21: Tip position x-coordinate of the spiral trajectory over time for different simulation models.

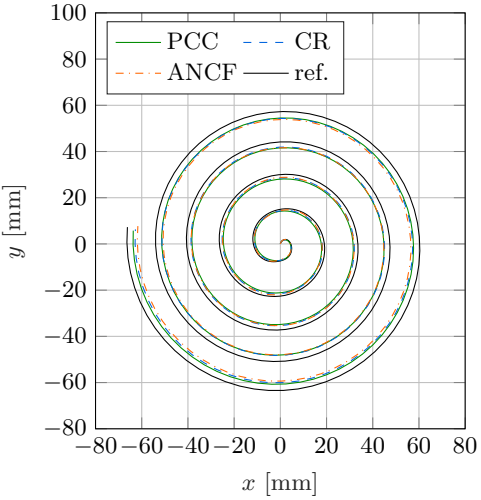


Figure 2.22: Tip position of the spiral trajectory for different simulation models.

given. In Tab. 2.3 the step sizes required to achieve this accuracy are given. For the variable step size solvers, the step size range used by the solver is given.

It can be seen that the ANCF model is clearly not real-time capable for any of

Table 2.2: Comparison of the computation time T on the spiral trajectory of 3 s length for different models, solvers and accuracies.

	`ode15s`	`ode45`	implicit Euler		
$e_{\max}$	$9.0{\cdot}10^{-4}$ mm	$9.0{\cdot}10^{-4}$ mm	10^{-2} mm	10^{-3} mm	10^{-4} mm
PCC	0.105 s	0.106 s	0.0633 s	0.263 s	2.35 s
CR	0.828 s	21.2 s	0.301 s	1.26 s	9.88 s
ANCF	9.69 s	57.9 s	24.0 s	40.6 s	377 s

Table 2.3: Comparison of the step width h on the spiral trajectory for different models, solvers and accuracies.

	`ode15s`	`ode45`	implicit Euler		
$e_{\max}$	$9.0{\cdot}10^{-4}$ mm	$9.0{\cdot}10^{-4}$ mm	10^{-2} mm	10^{-3} mm	10^{-4} mm
PCC	$2.9 \cdot 10^{-6} \ldots 9.5 \cdot 10^{-4}$ s	$2.9 \cdot 10^{-6} \ldots 9.5 \cdot 10^{-4}$ s	$1.6{\cdot}10^{-1}$ s	$2.4{\cdot}10^{-2}$ s	$2.0{\cdot}10^{-3}$ s
CR	$9.7 \cdot 10^{-7} \ldots 7.3 \cdot 10^{-4}$ s	$4.1 \cdot 10^{-7} \ldots 1.2 \cdot 10^{-2}$ s	$1.7{\cdot}10^{-1}$ s	$2.5{\cdot}10^{-2}$ s	$2.5{\cdot}10^{-3}$ s
ANCF	$5.4 \cdot 10^{-6} \ldots 2.5 \cdot 10^{-5}$ s	$3.4 \cdot 10^{-5} \ldots 2.1 \cdot 10^{-2}$ s	$1.7{\cdot}10^{-1}$ s	$3.0{\cdot}10^{-2}$ s	$2.0{\cdot}10^{-3}$ s

the considered solvers and step sizes. The PCC model is real-time capable for all considered solvers and step sizes, the CR model for all step sizes up to $2.5 \cdot 10^{-2}$ s. For comparable accuracy, the `ode15s` solver is always faster than the implicit Euler method, which is expected and can be explained by its variable step size and the higher order of the solver. The `ode45` solver is the only explicit solver considered. For the PCC model it shows a similar performance as the `ode15s` solver. For the other two models it is much slower. This can be explained by the fact that the PCC model only includes bending and torsion, which tend to have similar eigenfrequencies for typical soft robotic designs and thus result in a non-stiff or slightly stiff ODE, depending on the parameters of the model. The CR and ANCF models, on the other hand, also include strain and shear, which are typically much stiffer than bending and torsion. This results in a stiff ODE that can be solved much more efficiently by implicit solvers.

2.6 Tendon Actuation

Besides fluid actuation, tendon actuation is one of the most popular actuation methods for soft robots. Typically, one or more tendons with the indices $k = 1 \dots N_{\text{tendon}}$ run along the outside of a beam-shaped soft robot with a certain distance of the tendons to the neutral fiber of the beam. Pulling on the tendons induces forces and moments on the soft robot, typically resulting in a bending deformation. For the sake of simplicity, it is assumed that all tendons run along the entire length of the soft robot. However, the extension to tendons running along only parts of the soft robot is straightforward. In a first step, the mechanical realization of the tendon actuation used in this work is presented. In a second step, the forces acting on the soft robot due to tendon actuation are derived. As a third step, the length of the tendon inside the soft robot, which changes as the robot deforms, is determined. Finally, the integration of the tendon actuation into the equations of motion of the beam models derived in Secs. 2.1 to 2.3 is shown.

2.6.1 Mechanical Setup

In the following, a beam-shaped soft robot shown in Fig. 2.23 is considered, which can be subdivided into N_{seg} segments with the indices $j = 1 \dots N_{\text{seg}}$. In Fig. 2.24 a segment of the soft robot is shown in detail. For each tendon and piece there is a point $\mathrm{P}_{j,k,\text{in}}$ where the tendon with the index k enters the segment with the index j and, except for the last segment, a point $\mathrm{P}_{j,k,\text{out}}$ where this tendon leaves the segment.

For the derivation of the actuation forces, moments and tendon lengths, the position vectors $\boldsymbol{r}_{\mathrm{OQ}_j}$ and rotation matrices $\boldsymbol{R}_{\mathrm{OQ}_j}$ of the individual segments as well as the position vectors $\boldsymbol{r}_{\mathrm{OP}_{j,k,\text{in}}}$ and $\boldsymbol{r}_{\mathrm{OP}_{j,k,\text{out}}}$ of the end points of the tendon guides have to be determined. These quantities depend on the deformation of the soft robot and can be determined e.g. using one of the beam models described in chapter 2. Note that the discretization into N_{seg} segments for actuation does not necessarily have to be identical with the discretization into N_{pieces} pieces used for modeling with beam models presented in chapter 2. If this is not the case, an interpolation of the positions and rotations may be required.

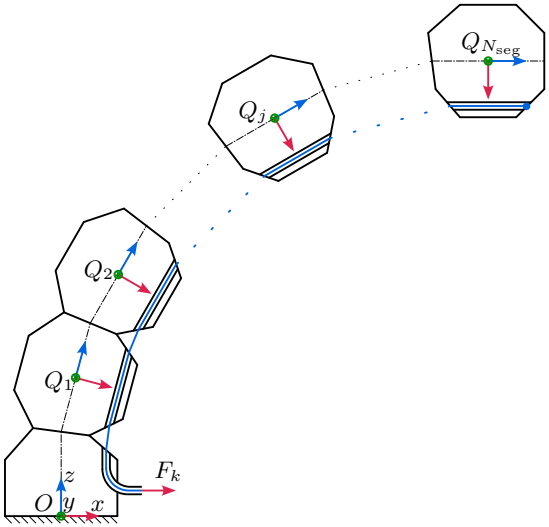


Figure 2.23: Discretization used for calculation of forces and moments resulting from tendon actuation.

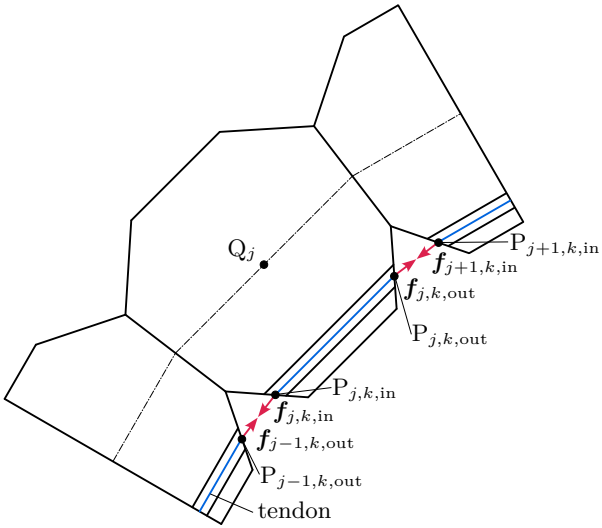


Figure 2.24: Forces resulting from tendon actuation for one segment.

2.6.2 Actuation Forces and Moments

To derive the tendon forces, the system is cut free. Without loss of generality, it can be assumed that the tendons are fixed in the tendon guides within the segments. For simplicity, belt friction is neglected, resulting in a constant magnitude of force F_k within each of the tendons. At the points $\mathrm{P}_{j,k,\mathrm{in}}$ and $\mathrm{P}_{j,k,\mathrm{out}}$ the incoming tendon forces $\boldsymbol{f}_{j,k,\mathrm{in}}$ and the outgoing tendon forces $\boldsymbol{f}_{j,k,\mathrm{out}}$ act. These forces point into the direction of the incoming or outgoing tendon, respectively and can be calculated as

$$\boldsymbol{f}_{j,k,\mathrm{in}} = F_k \frac{\boldsymbol{r}_{\mathrm{OP}_{j-1,k,\mathrm{out}}} - \boldsymbol{r}_{\mathrm{OP}_{j,k,\mathrm{in}}}}{\left|\left|\boldsymbol{r}_{\mathrm{OP}_{j-1,k,\mathrm{out}}} - \boldsymbol{r}_{\mathrm{OP}_{j,k,\mathrm{in}}}\right|\right|}, \tag{2.83}$$

$$\boldsymbol{f}_{j,k,\mathrm{out}} = \begin{cases} F_k \frac{\boldsymbol{r}_{\mathrm{OP}_{j+1,k,\mathrm{in}}} - \boldsymbol{r}_{\mathrm{OP}_{j,k,\mathrm{out}}}}{\left|\left|\boldsymbol{r}_{\mathrm{OP}_{j+1,k,\mathrm{in}}} - \boldsymbol{r}_{\mathrm{OP}_{j,k,\mathrm{out}}}\right|\right|} & j < N_{\mathrm{seg}} \\ \boldsymbol{0} & j = N_{\mathrm{seg}} \end{cases}. \tag{2.84}$$

Note that for the last segment, the outgoing force $\boldsymbol{f}_{j,k,\mathrm{out}}$ vanishes, because the tendon ends at the segment. Since the tendon forces act eccentrically, there are also resulting moments $\boldsymbol{\ell}_{\boldsymbol{j,k,\mathrm{in}}}$ and $\boldsymbol{\ell}_{\boldsymbol{j,k,\mathrm{out}}}$ which can be calculated as

$$\boldsymbol{\ell}_{\boldsymbol{j,k,\mathrm{in}}} = (\boldsymbol{r}_{\mathrm{OP}_{j,k,\mathrm{in}}} - \boldsymbol{r}_{\mathrm{OQ}_j}) \times \boldsymbol{f}_{j,k,\mathrm{in}} \tag{2.85}$$

$$\boldsymbol{\ell}_{\boldsymbol{j,k,\mathrm{out}}} = (\boldsymbol{r}_{\mathrm{OP}_{j,k,\mathrm{out}}} - \boldsymbol{r}_{\mathrm{OQ}_j}) \times \boldsymbol{f}_{j,k,\mathrm{out}}. \tag{2.86}$$

The forces $\boldsymbol{f}_j$ and moments $\boldsymbol{\ell}_j$ acting on segment j can then be calculated as

$$\boldsymbol{f}_j = \sum_{k=1}^{N_{\mathrm{tendon}}} \boldsymbol{f}_{j,k,\mathrm{in}} + \boldsymbol{f}_{j,k,\mathrm{out}}, \tag{2.87}$$

$$\boldsymbol{\ell}_j = \sum_{k=1}^{N_{\mathrm{tendon}}} \boldsymbol{\ell}_{j,k,\mathrm{in}} + \boldsymbol{\ell}_{j,k,\mathrm{out}}. \tag{2.88}$$

Note that the forces $\boldsymbol{f}_j$ and moments $\boldsymbol{\ell}_j$ depend linearly on the tendon forces F_k, which allows to write equations (2.87) and (2.88) in the form

$$\boldsymbol{f}_j = \boldsymbol{B}_{j,\mathrm{f}}(\boldsymbol{y})\boldsymbol{F} \tag{2.89}$$

$$\boldsymbol{\ell}_j = \boldsymbol{B}_{j,\ell}(\boldsymbol{y})\boldsymbol{F} \tag{2.90}$$

with the input distribution matrices $\boldsymbol{B}_{j,\mathrm{f}}(\boldsymbol{y})$ and $\boldsymbol{B}_{j,\ell}(\boldsymbol{y})$ which depend on the deformation of the robot only.

Tendon Length

In addition to the tendon forces, the tendon length s_k within the soft robot is also relevant for practical applications. The tendon length is important, for example, if position-controlled actuators are used instead of torque-controlled actuators.

The tendon length s_k can be divided into a geometric part $s_{\text{geo},k}$ and an elastic part $s_{\text{el},k}$, which results in

$$s_k = s_{\text{geo},k} + s_{\text{el},k}. \tag{2.91}$$

The geometric part $s_{\text{geo},k}$ results from the change in length of the tendon due to the change in geometry of the soft robot during deformation. It can be derived as

$$s_{\text{geo},k} = \sum_{j=1}^{N_{\text{seg}}} \left(\boldsymbol{r}_{\text{OP}_{j,k,\text{in}}} - \boldsymbol{r}_{\text{OP}_{j-1,k,\text{out}}} + \boldsymbol{r}_{\text{OP}_{j,k,\text{out}}} - \boldsymbol{r}_{\text{OP}_{j,k,\text{in}}} \right). \tag{2.92}$$

The elastic part $s_{\text{el},k}$ results from the elongation of the tendon due to the elasticity of the tendon. It can be calculated as

$$s_{\text{el},k} = \frac{F_k (s_{\text{geo},k} + s_{\text{ext},k})}{E_{\text{tendon}} A_{\text{tendon}}} \tag{2.93}$$

assuming linear-elastic material properties with the Young's modulus E_{tendon} and the cross section area A_{tendon} of the tendon as well as the constant length of the tendon $s_{\text{ext},k}$ between soft robot and actuator. For practical applications, the geometric length $s_{\text{geo},k}$ can be replaced by the tendon length in the unloaded straight reference configuration $s_{0,k}$, as the geometric length change $\Delta s_{\text{geo},k} = s_{\text{geo},k} - s_{0,k}$ is typically small compared to the tendon length $s_{0,k}$. This allows to rewrite equation (2.93) as

$$s_{\text{el},k} = F_k / c_k \tag{2.94}$$

with the stiffness constant c_k of the tendon, which usually has to be determined experimentally. The stiffness constants c_k can be assembled to the tendon stiffness matrix

$$\boldsymbol{C}_{\text{tendon}} = \text{diag}(c_1, \ldots, c_{N_{\text{tendon}}}). \tag{2.95}$$

Note that while the elastic part of the tendon length $s_{\text{el},k}$ can usually be neglected for quasi-static motion of the soft robot, it plays an important role for dynamic applications, where the tendon forces F_k are much larger.

2.6.3 Equations of Motion

Finally, the tendon actuation can be integrated into the equations of motion of the beam models in ODE form obtained in Secs. 2.1 to 2.3. This results in the equations of motion in control affine form

$$\boldsymbol{M}(\boldsymbol{y}, t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) = \boldsymbol{q}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{F} \tag{2.96}$$

with the input distribution matrix

$$\boldsymbol{B}(\boldsymbol{y}) = \sum_{j=1}^{N_{\text{pieces}}} \boldsymbol{J}_{\text{T}}^{\top}(\boldsymbol{y})\boldsymbol{\Theta}_{\text{T}}\boldsymbol{B}_{j,\text{f}}(\boldsymbol{y}) + \boldsymbol{J}_{\text{R}}^{\top}(\boldsymbol{y})\boldsymbol{\Theta}_{\text{R}}\boldsymbol{B}_{j,\ell}(\boldsymbol{y}) \tag{2.97}$$

and the Jacobians of translation $\boldsymbol{J}_{\text{T}}^{\top}$ and rotation $\boldsymbol{J}_{\text{R}}^{\top}$ of the beam models. The distribution matrices $\boldsymbol{\Theta}_{\text{T}} \in \mathbb{R}^{N_{\text{pieces}} \times N_{\text{seg}}}$ of the forces and $\boldsymbol{\Theta}_{\text{R}} \in \mathbb{R}^{N_{\text{pieces}} \times N_{\text{seg}}}$ of the moments describe the distribution of the forces and moments acting on the segments to the pieces used as discretization for the beam models. For $N_{\text{pieces}} = N_{\text{seg}}$ the distribution matrices $\boldsymbol{\Theta}_{\text{T}}$ and $\boldsymbol{\Theta}_{\text{R}}$ simplify to the identity matrix $\boldsymbol{I}$.

If instead of the tendon forces $\boldsymbol{F}$, the tendon lengths $\boldsymbol{s}$ are considered as system input, using equations (2.91) and (2.94), the equation of motion (2.96) can be rewritten as

$$\boldsymbol{M}(\boldsymbol{y}, t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) = \tilde{\boldsymbol{q}}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) + \tilde{\boldsymbol{B}}(\boldsymbol{y})\boldsymbol{s} \tag{2.98}$$

where

$$\tilde{\boldsymbol{q}}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) = \boldsymbol{q}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) - \boldsymbol{C}_{\text{tendon}}\boldsymbol{s}_{\text{geo}}(\boldsymbol{y}), \tag{2.99}$$

$$\tilde{\boldsymbol{B}}(\boldsymbol{y}) = \boldsymbol{B}(\boldsymbol{y})\boldsymbol{C}_{\text{tendon}}. \tag{2.100}$$

Note that the control affine form of the equations of motion is preserved also in this case.

Data-Driven Models

Data-driven models play an important role in soft robotics because they allow for the direct incorporation of manufacturing inaccuracies and other effects that are difficult to model [Della SantinaEtAl23]. However, the amount of training data required often limits the applicability, especially for dynamic models.

There is a wide range of different data-driven models for soft robots in the literature. One way to classify them is to distinguish between quasi-static and dynamic models. For the quasi-static modeling of soft robots, feedforward neural networks (FNNs) [WieseEtAl19, FangEtAl22] are widely used. Besides FNNs, statistical methods such as support vector regression [MelinguiEtAl18] and dynamic Gaussian mixture models [YuEtAl19] are also used. Compared to dynamic models, quasi-static models are typically less complex and require much less training data. Dynamic modeling, however, is more challenging and thus requires more sophisticated methods. For the dynamic modeling of soft robots, a variety of different machine learning models are used, such as neural network based nonlinear autoregressive networks (NARX) [ThuruthelEtAl17b] and recurrent neural networks (RNNs) [ChenEtAl24a]. In addition, specialized network topologies such as D-extended un-parallel Prandtl–Ishlinskii model (D-EUPI) networks for hysteresis compensation [ZhangEtAl19] are used to address specific issues. In addition, also statistical methods such as locally weighted projection regression [LeeEtAl17b] are used. Furthermore, subspace methods based on the Koopman operator theory [BruderEtAl21] or spectral submaifold reduction [AloraEtAl23] are used, which allow to derive models in a particularly simple form on a well-chosen subspace. A more detailed overview of data-driven models used in soft robotics can e.g. be found in [ChenEtAl24b, Della SantinaEtAl23].

In the following, first, in Sec. 3.1 a soft robotic test system is presented. In this chapter, this test system is used to test the data-driven models. Then, in Secs. 3.2 and 3.3, a quasi-static and a dynamic data-driven model for this test system are presented and evaluated in terms of their accuracy, computational efficiency, and data requirements.

3.1 Test System

For the evaluation of the performance of data-driven models a test system is required. In the following, first the design of the test system and then the treatment of the redundant actuation is discussed. This test system is also used in chapter 4 for the experimental evaluation of the sensor performance and in chapter 5 for the experimental evaluation of the feedforward control performance.

3.1.1 Design of the Test System

In this work, a beam-shaped soft robot shown in Fig. 3.1 is considered as a test system. It can bend in x- and y-direction and is sufficiently stiff in the longitudinal direction, such that elongation can be neglected. This results in the tip of the soft robot moving on a surface that can be approximated by a hemisphere. Thus, the tip position $\boldsymbol{z}$, which is considered as the system output in the following, can be fully described by its x- and y-coordinate. The workspace of the soft robot is shown in Fig. 3.2.

The soft robot has a length of 202 mm and a radius of 30 mm. It is made of silicone of the type "HT45" [Zhermack SpA13] in a silicone molding process. The soft robot is redundantly actuated by three nylon tendons with a diameter of 0.45 mm via three servos of the type Hitec "HS-311" [Hitec Commercial Solutions22]. In a second version of this test rig, Chaservo "DS20" [AKATJA GmbH23] servos are used instead, which have a higher positioning accuracy, higher angular velocity and higher torque. The tendons are evenly distributed around the circumference of the soft robot, as shown in Fig. 3.3. The length change $\boldsymbol{s} = \begin{bmatrix} s_1 & \dots & s_3 \end{bmatrix}^\top$ of the tendons towards the straight reference configuration of the soft robot is directly controlled by the servos and is considered as the system input. For simplicity, it is referred to as the tendon length in the following. Especially for modeling of the soft robot with mechanical models, such as the beam models presented in chapter 2, it makes sense to consider the tendon forces $\boldsymbol{F} = \begin{bmatrix} F_1 & \dots & F_3 \end{bmatrix}^\top$ instead of the tendon length $\boldsymbol{s}$ as system input, even though the tendon forces $\boldsymbol{F}$ cannot be controlled directly. Equations (2.91) to (2.93) can be used to derive a relationship between the tendon length $\boldsymbol{s}$ and the tendon forces $\boldsymbol{F}$. For accurate control of the soft robot, it is important that the tendons are always slightly pretensioned to reduce dead band and hysteresis effects.

The redundancy in actuation is mainly due to the elasticity of the tendons and the high stiffness of the soft robot in the longitudinal direction. For any sufficiently large sum of tendon forces $\hat{F} = \sum F_k$ and any tip position $\boldsymbol{z}$ in the workspace, a set of tendon forces $\boldsymbol{F}$ can be found to reach that position. Here $k = 1 \dots 3$ is the index of the tendon. Due to the elasticity of the tendons, and to some

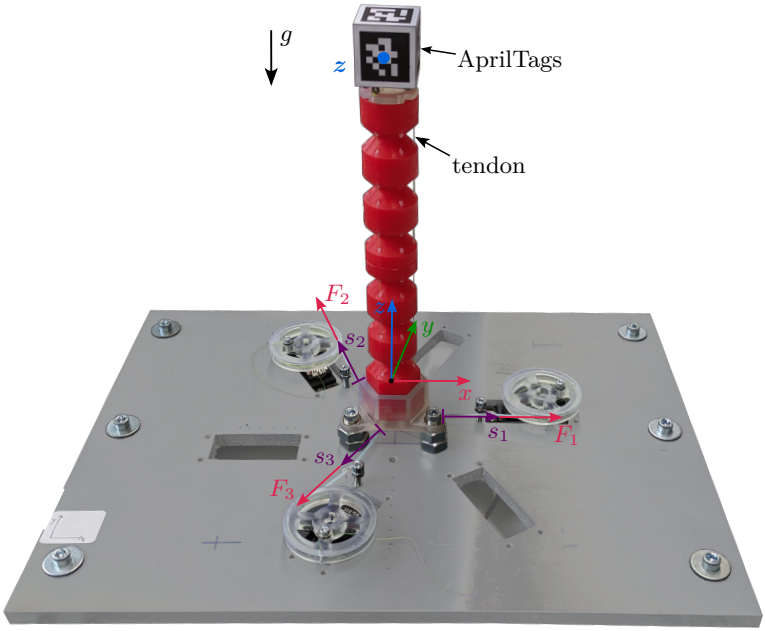


Figure 3.1: Soft robotic test system.

extent the elasticity of the soft robot in the longitudinal direction, this also leads to different tendon lengths $\boldsymbol{s}$. Note that while the elasticity of the soft robot in longitudinal direction has some effect on the tendon length s_k, which should be considered, its effect on the tip position $\boldsymbol{z}$ can be neglected. Overall, the system can be considered as a soft version of an inverse pendulum stabilized by the tendon forces F_k. Note, however, that due to the continuous deformation, the system behavior is more complex than for its rigid counterpart.

For camera tracking of the tip position $\boldsymbol{z}$, a cube with AprilTag [Olson11] fiducial markers is attached to the tip of the robot. This allows the spatial position and rotation of the tip to be determined with a single camera. The camera used is a webcam of the type Logitech "Brio Ultra HD Pro" with a frame rate of up to 90 fps. As an alternative to the cube, a spherical reflective marker can be attached to the tip and can be tracked with a single Optitrack "Prime 17W" camera. This system provides a more accurate and computationally efficient measurement of the tip position with a high frame rate of up to 360 fps, but is limited to 2D measurements in pixel coordinates since only one camera is available.

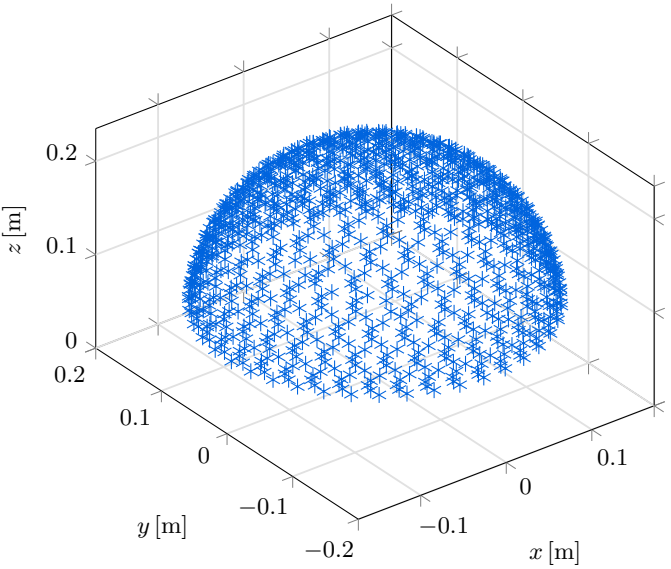


Figure 3.2: Workspace of the soft robotic test system.

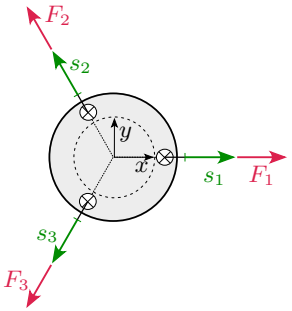


Figure 3.3: Horizontal cut through the soft robot, showing the arrangement of the tendons of the soft robot.

3.1.2 Resolving the Redundant Actuation

For control problems, redundantly actuated systems pose a great challenge because there is no unique solution for the system inputs. Therefore, additional constraints must be introduced. In the case of trajectory tracking control, these are often smoothness constraints, which are not always easy to define. In addition, redundantly actuated systems are typically more complex than their

non-redundantly actuated counterparts, leading to higher data requirements for data-driven modeling.

For the systems presented in this section, the redundancy of the actuation can be removed by introducing a reduced set of allowed inputs without affecting the size of the workspace or the smoothness of the trajectories within the workspace. For this, in a first step, a reduced system input $\boldsymbol{F}_{\mathrm{red}} = \begin{bmatrix} F_{\mathrm{red},x} & F_{\mathrm{red},y} \end{bmatrix}^\top$ with

$$\underbrace{\begin{bmatrix} F_{\mathrm{red},x} \\ F_{\mathrm{red},y} \end{bmatrix}}_{\boldsymbol{F}_{\mathrm{red}}} = \underbrace{\begin{bmatrix} 1 & -\sin(30^\circ) & -\sin(30^\circ) \\ 0 & \cos(30^\circ) & -\cos(30^\circ) \end{bmatrix}}_{\boldsymbol{\Phi}} \underbrace{\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}}_{\boldsymbol{F}} \tag{3.1}$$

is defined. Here, $\boldsymbol{\Phi}$ is a projection matrix and the reduced system input $\boldsymbol{F}_{\mathrm{red}}$ can be interpreted as the projection of the three tendon forces $\boldsymbol{F}$ in x- and y-direction. The reduced system input $\boldsymbol{F}_{\mathrm{red}}$ is visualized in Fig. 3.4. For practical applications, the inverse of this projection is often needed as well. Since the projection matrix $\boldsymbol{\Phi}$ is rank deficient, the inverse of the projection is not unique. When selecting a projection in inverse direction, it is important to make sure that the tendon forces $\boldsymbol{F}$ are always positive, since tendons can only transmit tensile forces. In addition, from a practical point of view, it makes sense to always keep the tendons slightly pretensioned to reduce dead band and hysteresis effects, and to choose tendon forces that are not unnecessarily large to reduce wear. A possible choice for the projection in inverse direction is

$$\boldsymbol{F} = \boldsymbol{\Phi}^\dagger \boldsymbol{F}_{\mathrm{red}} + F_{\mathrm{pre}} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top \tag{3.2}$$

with the pseudoinverse

$$\boldsymbol{\Phi}^\dagger = \begin{bmatrix} \boldsymbol{\Phi}_1^\dagger & \boldsymbol{\Phi}_2^\dagger \end{bmatrix}, \tag{3.3}$$

$$\boldsymbol{\Phi}_1^\dagger = \begin{cases} \begin{bmatrix} 0 & -1/\sin(30^\circ) & -1/\sin(30^\circ) \end{bmatrix}^\top & F_{\mathrm{red},x} \leq 0 \\ \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top & \text{otherwise,} \end{cases} \tag{3.4}$$

$$\boldsymbol{\Phi}_2^\dagger = \begin{cases} \begin{bmatrix} -1/\sin(30^\circ) & 0 & -1/\cos(30^\circ) \end{bmatrix}^\top & F_{\mathrm{red},y} \leq 0 \\ \begin{bmatrix} 1/\sin(30^\circ) & 1/\cos(30^\circ) & 0 \end{bmatrix}^\top & \text{otherwise} \end{cases} \tag{3.5}$$

and the minimum pretensioning force F_{pre} for each of the tendons. With the inverse projection (3.2) the reduced set $\mathcal{F}_{\mathrm{red}}$ of tendon forces can be defined as $\mathcal{F}_{\mathrm{red}} = \{\boldsymbol{F} : \boldsymbol{F} = \boldsymbol{\Phi}^\dagger \boldsymbol{F}_{\mathrm{red}} + F_{\mathrm{pre}} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top \mid \boldsymbol{F}_{\mathrm{red}} \in \mathbb{R}^2, F_1, \dots F_3 \leq F_{\max}\}$ where $F_{\max}$ is the maximum possible tendon force of the system.

For the application to the test system, the tendon lengths are usually considered as the system input instead of the tendon forces. Here, a reduced set $\mathcal{S}_{\mathrm{red}}$ of tendon lengths $\boldsymbol{s}$ corresponding to the reduced set $\mathcal{F}_{\mathrm{red}}$ of tendon forces in the quasi-static case can be determined. To do this, a quasi-static version of one of the

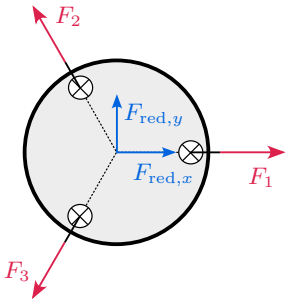


Figure 3.4: Projection of the tendon forces in axial direction to resolve the redundancy of the actuation.

beam models presented in chapter 2 is used to simulate the system behavior on the reduced set $\mathcal{F}_{\mathrm{red}}$ of tendon forces. Then, using equation (2.91), the tendon length $\boldsymbol{s}$ corresponding to the tendon forces $\boldsymbol{F}$ in the reduced set $\mathcal{F}_{\mathrm{red}}$ of tendon forces can be calculated from the simulation results. For practical applications, the set $\mathcal{S}_{\mathrm{red}}$ can be approximated by the set $\tilde{\mathcal{S}}_{\mathrm{red}} = \{\boldsymbol{s} : s_1 + s_2 + s_3 = 0 \mid s_1 \dots s_3 \leq s_{\max}\}$, where $s_{\max}$ is the maximum possible tendon length change. This set forms a plane in the three-dimensional space of tendon forces $s_1 \dots s_3$. The reduced sets $\mathcal{S}_{\mathrm{red}}$ and $\tilde{\mathcal{S}}_{\mathrm{red}}$ of tendon length can also be used for dynamic applications. Note, however, that for dynamic applications the usage of the set $\mathcal{S}_{\mathrm{red}}$ of reduced tendon length leads to slightly different results than the usage of the reduced set $\mathcal{F}_{\mathrm{red}}$ of tendon forces as system inputs. For practical applications, however, this difference is typically not of importance.

3.2 Neural Network based Quasi-Static Model

For quasi-static modeling of soft robots feedforward neural networks can be used. In this work, the test system presented in Sec. 3.1 is considered as an example. The inputs $\boldsymbol{s}$ of the test system are restricted to the reduced set $\mathcal{S}_{\mathrm{red}}$ of tendon length to resolve the redundancy of the test system as explained in Sec. 3.1.2. In the following, first, the structure of the neural network used to model the system is described. Second, the training data collection process is described. Third, the training process is described and the choice of hyperparameters of the neural network is investigated. Finally, the performance of the NN-based quasi-static model is evaluated regarding its accuracy and computational efficiency.

3.2.1 Structure of the Neural Network

The structure of the neural network used to model this system is shown in Fig. 3.5. The input to the neural network is the tendon length $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^\top$, the output is the tip position $\boldsymbol{z} = \begin{bmatrix} z_x & z_y \end{bmatrix}^\top$. The system input $\boldsymbol{s}$ is restricted to the set S_{red} of reduced system inputs introduced in Sec. 3.1.2 so that the redundant actuation of the system is resolved. This reduces the amount of training data required by reducing the complexity of the system, and simplifies the application of the model to control problems by allowing unique inversion. The NN has n_{layer} hidden layers with n_{neuron} neurons each. Linear activation functions are used in the output layer, and tanh activation functions in the hidden layers. The NN is implemented in MATLAB using the "Deep Learning Toolbox". After training, the NN is exported as a function and compiled as a MATLAB `mex` function to improve the computational efficiency. The choice of hyperparameters of the NN is examined in Sec. 3.2.3.

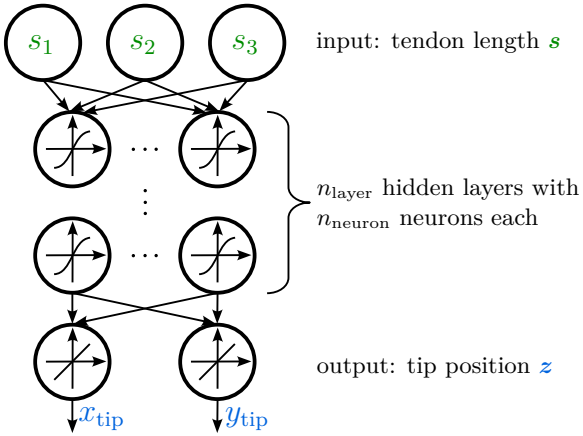


Figure 3.5: Structure of the neural network used to model the quasi-static forward behavior.

3.2.2 Training Data Collection

For training data collection, in a first step a set of tip position points $\boldsymbol{z}_{\mathrm{PCC}}$ evenly distributed over the workspace with corresponding tendon length $\boldsymbol{s}_{\mathrm{PCC}} \in S_{\mathrm{red}}$ is derived using a quasi-static version of the PCC model presented in Sec. 2.1. The PCC model is set up with nominal parameters of the soft robot derived from a CAD model to provide an initial estimate of the soft robot behavior. Direct use of the PCC model data for training is possible and captures the basic system behavior, but omits real-world discrepancies due to factors such as manufacturing inaccuracies. Therefore, real-world measurements are essential for accurate modeling of the physical soft robot. Nevertheless, the PCC model data, is needed to ensure that the system inputs remain within the set S_{red} of reduced system inputs during training data collection on the physical system. This ensures that the robot is not damaged and that the resulting NN model does not contain redundant actuation.

In a second step, the calculated values for the tendon length $\boldsymbol{s}_{\mathrm{PCC}}$ are used to approach each of the tip position points in turn with the physical robot. For each of the points, the actual position of the tip position $\boldsymbol{z}_{\mathrm{meas}}$ is measured using the AprilTag based camera tracking system. Finally, the values of the calculated tendon length $\boldsymbol{s}_{\mathrm{PCC}}$ and the measured tip position $\boldsymbol{z}_{\mathrm{meas}}$ are evenly split into a training set containing 70 % of the data, a validation set containing 15 % of the data and a test set containing the remaining 15 % of the data. The training set is used for the actual training of the neural network, the validation set is used for supervising the training process, for early stopping during the training process to prevent overfitting, and for tuning the hyperparameters. The test set is used to finally evaluate the performance of the neural network.

In Fig. 3.6 an excerpt of the collected data is shown. The projection into the $x-y$-plane of the tip position points $\boldsymbol{z}_{\mathrm{PCC}}$ computed with the PCC model and the measured tip position points $\boldsymbol{z}_{\mathrm{meas}}$ is shown. It can be seen that the overall behavior of the physical robot and the PCC model agrees well. However, there are differences between the physical robot and the model that need to be taken into account for accurate modeling. These differences are mainly due to the fact that the CAD model does not perfectly represent the physical robot, e.g. due to manufacturing inaccuracies. With some parameter tuning, the accuracy of the PCC model can usually be greatly improved. However, in soft robotics, typically some effects remain that are difficult to parameterize and therefore difficult to include in the model. Since the PCC model is only used to obtain consistent values for the tendon length $\boldsymbol{s}_{\mathrm{PCC}}$ used to collect training data, no parameter tuning is required.

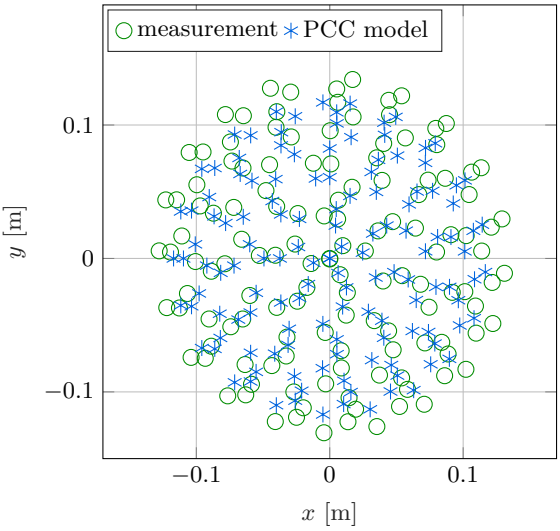


Figure 3.6: Tip position $\boldsymbol{z}$ from the nominal PCC model and from the measurement collected for training of the neural network.

3.2.3 Hyperparameters of the Neural Network

The choice of hyperparameters is crucial for a good performance of the neural network. The required number of hidden layers n_{layer} and the number of neurons n_{neuron} in each of these layers depends on the complexity of the quasi-static behavior of the soft robot and must be determined heuristically. Additionally, the number of training samples $n_{\text{samples,train}}$ needed to train the neural network is an important parameter that determines the overall training effort. These three parameters are examined in the following.

In a first step, a dataset of 926 samples, evenly distributed over the workspace, is collected as described in Sec. 3.2.2. Then different networks with $n_{\text{layer}} = 1 \dots 5$ hidden layers and $n_{\text{neuron}} = 1 \dots 50$ neurons per layer are trained and compared regarding the root mean square error e_{RMS} of the neural network output, which is the tip position. It is defined as

$$e_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{z}_{\text{sim}} - \boldsymbol{z}_{\text{meas}}||^2}, \tag{3.6}$$

where N is the number of samples, $\boldsymbol{z}_{\text{meas}}$ is the measured tip position and $\boldsymbol{z}_{\text{sim}}$ is the tip position obtained from the data-based simulation model. To account for random effects, such as the random initialization of the weights of the neural

network before training, the median value from 10 training runs is considered below for each set of hyperparameters. The results are shown in Fig. 3.7. It can be seen that the best performance can be achieved with $n_{\text{neuron}} \approx 10$ neurons per layer. For less than $n_{\text{neuron}} = 4$ neurons per layer the error e_{RMS} strongly increases, which can be explained by underfitting. For more than $n_{\text{neuron}} = 10$ neurons per layer the error e_{RMS} does not further decrease. Note that due to the early stopping during training, no increase of the error for very large networks can be observed, as training is stopped before overfitting can occur. The number of hidden layers n_{layer} has hardly any influence. The only notable difference is that for a medium number of $4 \dots 15$ neurons per layer, the networks with one hidden layer perform slightly worse than the other networks. Therefore, in the following a network with $n_{\text{layer}} = 2$ hidden layers and $n_{\text{neuron}} = 10$ neurons per layer is used.

Finally, the required size of the training set is examined. Therefore, the neural network with $n_{\text{layer}} = 2$ hidden layers and $n_{\text{neuron}} = 10$ neurons per layer is trained with different numbers of samples of $n_{\text{sample}} = 1 \dots 646$ in the training set. The results are plotted in Fig. 3.8. It can be seen that $n_{\text{samples}} \approx 80$ samples in the training set are needed to achieve a tip position error of less than 1 mm. For more than $n_{\text{samples}} \approx 150 \dots 200$ samples the error only slightly drops. In the following, a training set of 150 samples is used. Together with the validation set and the test set, a total of ≈ 215 data points are required, assuming a split of 70 % of the data in the training set and 15 % each of the data in the validation and test set. On the physical robot this data can be collected in about 5 min.

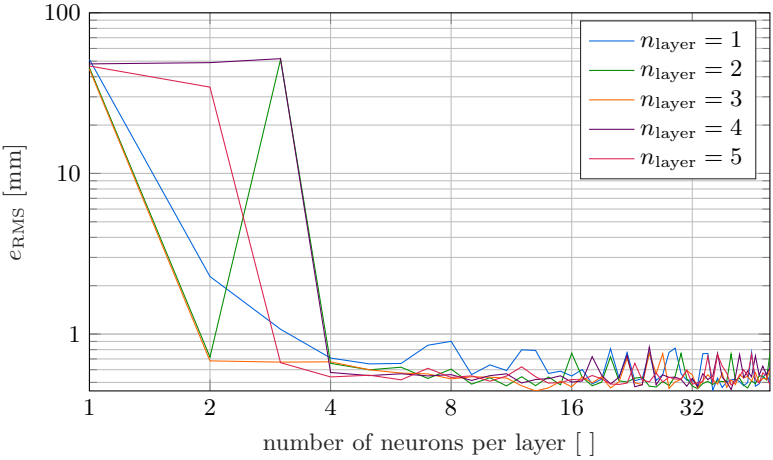


Figure 3.7: Influence of the number of layers n_{layer} and the number of neurons per layer n_{neurons} on the accuracy of the NN quasistatic forward model.

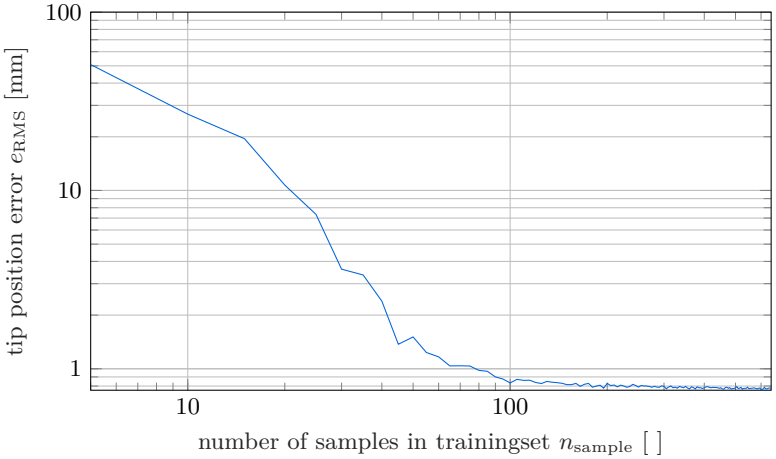


Figure 3.8: Influence of the training set size $n_{\mathrm{samples,train}}$ on the accuracy of the NN quasistatic forward model.

3.2.4 Model Performance

To evaluate the performance of the quasi-static NN model, a neural network with $n_{\mathrm{layer}} = 2$ hidden layers with $N_{\mathrm{neuron}} = 10$ neurons each is trained using a training set of $n_{\mathrm{sample}} = 150$ samples and a validation set of 32 samples. As a test set, the full test set collected in Sec. 3.2.3 with 139 samples is used.

In Fig. 3.9 the measured and simulated tip positions of the test set are plotted in a $x - y$ plot. A root mean square error of $e_{\mathrm{RMS}} = 0.77\,\mathrm{mm}$ with a standard deviation of 0.86 mm was achieved on the test set, which is less than 0.30 % of the workspace diameter. The error is mainly due to a slight hysteretic behavior of the soft robot, resulting from cable friction in the tendon guides, which cannot be represented in this type of model. In addition, the error is in the order of the accuracy of the camera tracking system used. There are a few outliers with an error of up to 9.3 mm, which can be attributed to disturbances during the measurement recording. Overall, a very good accuracy was achieved, which is probably sufficient for most soft robotic applications. The main limitation of this model is that it does not include dynamics, which is only an issue if fast motions are required.

Finally, the computational efficiency of the neural network is evaluated. For this, the test set is evaluated by the neural network sample by sample 10 000 times, resulting in an average evaluation time of $T_{\mathrm{comp}} = 2.0\,\mu\mathrm{s}$ per sample for the compiled version of the neural network on a computer with an “Intel Core i7-1280P” CPU. Without compilation, the average evaluation time increases to a

value of $T_{\text{uncomp}} = 14.9\,\mu\text{s}$. In both cases, the computation time is several orders of magnitude shorter than typically required for practical applications.

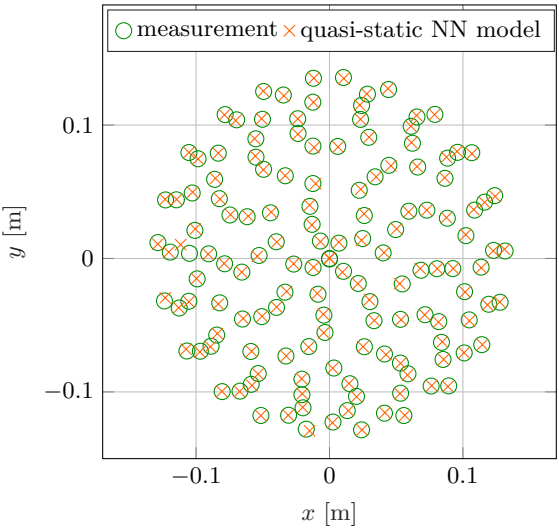


Figure 3.9: Tip position measurement and simulation using the quasi-static NN model on the test set.

3.3 Spectral Submanifold Reduction

Spectral submanifold reduction (SSMR) enables data-efficient, data-based learning of low-dimensional dynamics. It belongs to a class of subspace methods which aim to simplify the learning process of complex nonlinear dynamic systems by splitting the learning process of the dynamics into two steps. In a first step a projection to and from a submanifold is learned, in which a simpler description of the dynamics is possible. In a second step, the dynamics on the submanifold are learned. Besides SSMR [HallerPonsioen16], also Koopmann operator based methods, see e.g. [KomenoEtAl22] and latent space methods, see e.g. [LiuEtAl24, StölzleDella Santina24] belong to this class of methods.

In the following, first, the concept of spectral submanifolds is introduced. Then the data-based learning of a reduced order model is described. The derivation of the SSMR model follows [HallerPonsioen16, AloraEtAl23]. Finally, the application to the test system introduced in Sec. 3.1 is described and the performance of the SSMR model is evaluated. First results have been presented in [Duske24].

3.3.1 Spectral Submanifolds

Spectral submanifolds (SSMs) can be used for the description of systems in control linear form where the magnitude of the control inputs is moderate compared to the autonomous dynamics of the system. As shown in Sec. 2.6, tendon-actuated beam-shaped soft robots can usually be modeled with a system in control affine form. Additionally, the control inputs are often moderate compared to the autonomous dynamics. This class of systems can be written as a first order ODE of dimension n_f with m inputs in the form

$$\dot{\boldsymbol{x}}_\mathrm{f}(t) = \boldsymbol{A}_\mathrm{f}\boldsymbol{x}_\mathrm{f}(t) + \boldsymbol{f}_{\mathrm{nl,f}}(\boldsymbol{x}_\mathrm{f}(t)) + \epsilon\boldsymbol{B}_\mathrm{f}\boldsymbol{u}(t), \qquad 0 < \epsilon \ll 1 \tag{3.7}$$

where $\boldsymbol{x}_\mathrm{f} \in \mathbb{R}^{n_\mathrm{f}}$ is the state vector, $\boldsymbol{A}_\mathrm{f} \in \mathbb{R}^{n_\mathrm{f}\times n_\mathrm{f}}$ is the constant negative-definite and diagonalizable system matrix, $\boldsymbol{B}_\mathrm{f} \in \mathbb{R}^{n_\mathrm{f}\times m}$ is the constant input distribution matrix and $\boldsymbol{u} \in \mathbb{R}^m$ is the system input. The subscript f denotes the full state, as opposed to the reduced state $\boldsymbol{x}$, which is introduced in Sec. 3.3.2. The term $\boldsymbol{f}_{\mathrm{nl,f}}(\boldsymbol{x}_\mathrm{f}(t))$ summarizes the nonlinearities of the system and must satisfy $\boldsymbol{f}_{\mathrm{nl,f}}(\boldsymbol{0}) = \boldsymbol{0}$, $\partial\boldsymbol{f}_{\mathrm{nl,f}}(\boldsymbol{0})/\partial\boldsymbol{x}_\mathrm{f} = \boldsymbol{0}$. The parameter $0 < \epsilon \ll 1$ introduces the assumption that the magnitude of the control inputs is moderate compared to the autonomous dynamics. Additionally, it is assumed that $\boldsymbol{x}_\mathrm{f}(\boldsymbol{0})$ is an asymptotically stable equilibrium.

In a first step, the linear unperturbed part

$$\dot{\boldsymbol{x}}_\mathrm{f} = \boldsymbol{A}_\mathrm{f}\boldsymbol{x}_\mathrm{f} \tag{3.8}$$

of the system (3.7) is considered. The system matrix $\boldsymbol{A}_\mathrm{f}$ has n_f eigenvalues $\lambda_1 \dots \lambda_{n_\mathrm{f}}$ with $\mathrm{Re}(\lambda_1) \geq \mathrm{Re}(\lambda_2) \geq \dots \geq \mathrm{Re}(\lambda_{n_\mathrm{f}})$ and corresponding eigenvectors $\boldsymbol{e}_1 \dots \boldsymbol{e}_{n_\mathrm{f}}$. For each distinct eigenvalue λ_j there exists a real eigenspace $E_j \subset \mathbb{R}^{n_\mathrm{f}}$ spanned by the corresponding eigenvectors, which is also an invariant subspace of the linear system (3.8). Also, a spectral subspace E spanned by any combination of eigenspaces E_j is an invariant subspace of system (3.8). By the spectral mapping theorem [Harte23], any trajectory that starts in E will remain in E.

When nonlinearities are introduced, system (3.8) is extended to

$$\dot{\boldsymbol{x}}_\mathrm{f} = \boldsymbol{A}_\mathrm{f}\boldsymbol{x}_\mathrm{f} + \boldsymbol{f}_{\mathrm{nl,f}}(\boldsymbol{x}_\mathrm{f}(t)) \tag{3.9}$$

and superposition is lost. It follows that system (3.9) is not invariant on E. To regain invariance, the autonomous SSM $\mathcal{W}(E)$ is introduced, which is an invariant manifold of the autonomous nonlinear system (3.9) such that $\mathcal{W}(E)$ is tangent to E at the origin, has the same dimension as E and is strictly smoother than any other invariant manifold satisfying the prior conditions. The parametrization of $\mathcal{W}(E)$ can e.g. be expressed by polynomials. For model order reduction slow SSMs, which are constructed from the k slowest modes E_k are most important as nearby full system trajectories converge to them exponentially fast. In Fig. 3.10 an autonomous trajectory converging to the equilibrium $\boldsymbol{x}_\mathrm{e}$ on a SSM $\mathcal{W}(E)$ is visualized.

If the autonomous system (3.9) is extended to the full non-autonomous system (3.7), the invariance is lost again. However, the SSM $\mathcal{W}(E)$ is still relevant if ϵ is sufficiently small. Detailed results on the existence and relevance of SSMs for non-autonomous systems are given e.g. in [HallerPonsioen16].

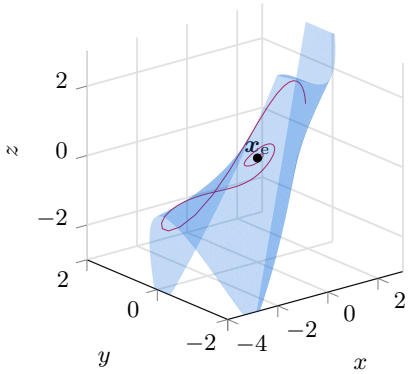


Figure 3.10: Autonomous trajectory converging to the equilibrium $\boldsymbol{x}_\mathrm{e}$ on a spectral submanifold $\mathcal{W}(E)$.

3.3.2 Learning the Reduced Order Spectral Submanifold

In the following, based on the theoretical concept of SSMs introduced in Sec. 3.3.1, the data-based learning of a reduced-order system of dimension n of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{f}_{\text{nl}}(\boldsymbol{x}(t)) + \epsilon \boldsymbol{B}\boldsymbol{u}(t), \qquad 0 < \epsilon \ll 1 \tag{3.10}$$

which approximates the full system (3.7), is described. Here $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the system matrix, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ is the input distribution matrix, $\boldsymbol{u} \in \mathbb{R}^m$ is the system input and $\boldsymbol{f}_{\text{nl}}(\boldsymbol{x}(t))$ summarizes the nonlinearities of the reduced system. First, the representation of the required data using the Takens delay embedding theorem and the data collection process for learning the reduced SSM are described. Then, the learning of the SSM and the input distribution matrix is described.

Takens Delay Embedding Theorem

For the data-based learning of a reduced SSM model, a sufficiently rich dataset $\boldsymbol{\mathcal{Y}}$ containing measurements of the full system state vector $\boldsymbol{x}_{\text{f}}$ over time of autonomous trajectories is required. However, measuring all states of a mechanical system is typically not possible. To overcome this issue Takens delay embedding Theorem [Takens81, SauerEtAl91] can be exploited to construct a state vector $\boldsymbol{y}(t) = [\tilde{\boldsymbol{y}}^\top(t), \tilde{\boldsymbol{y}}^\top(t+\Delta t), \tilde{\boldsymbol{y}}^\top(t+2\Delta t), \dots]^\top \in \mathbb{R}^p$ of dimension $p \geq 2n+1$ out of measurements $\tilde{\boldsymbol{y}}(t)$ taken at different points in time with the sample time Δt. For practical applications suitable values for p and Δt have to be determined experimentally, some theoretical guarantees are discussed in [SauerEtAl91].

Training Data Collection

For data collection, in a first step several autonomous trajectories with varying initial conditions are recorded with sample time ΔT. The autonomous trajectories are required to converge to the equilibrium $\boldsymbol{x} = \boldsymbol{0}$. Then snapshots of their state vector $\boldsymbol{y} \in \mathbb{R}^p$ at different times are stored in the matrix $\boldsymbol{\mathcal{Y}}_i$ which is defined as

$$\boldsymbol{\mathcal{Y}}_i = \begin{bmatrix} \boldsymbol{y}(t) & \boldsymbol{y}(t+\Delta T) & \boldsymbol{y}(t+2\Delta T) & \dots \end{bmatrix}. \tag{3.11}$$

Several of these trajectory datasets $\boldsymbol{\mathcal{Y}}_i$ are then stacked together to form the full dataset $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{p\times N_{\text{samples}}}$ defined as

$$\boldsymbol{\mathcal{Y}} = \begin{bmatrix} \boldsymbol{\mathcal{Y}}_1 & \boldsymbol{\mathcal{Y}}_2 & \boldsymbol{\mathcal{Y}}_3 & \dots \end{bmatrix} \tag{3.12}$$

where N_{samples} is the number of samples in the full dataset.

Learning of the Reduced SSM

The SSM $\mathcal{W}(E)$ can be described by a pair of smooth invertible transformation functions $\boldsymbol{y} = \boldsymbol{w}(\boldsymbol{x})$, $\boldsymbol{x} = \boldsymbol{v}(\boldsymbol{y})$, where $\boldsymbol{w}(\boldsymbol{x})$ uniquely maps the reduced

state $\boldsymbol{x} \in \mathbb{R}^n$ on the SSM $\mathcal{W}(E)$ to the observed state $\boldsymbol{y} \in \mathbb{R}^p$ and $\boldsymbol{v}(\boldsymbol{y})$ maps the observed state $\boldsymbol{y} \in \mathbb{R}^p$ to the reduced state $\boldsymbol{x} \in \mathbb{R}^n$. Due to the invariance and tangency properties of the SSM, the two maps must satisfy the invertibility relations $\boldsymbol{y} = \boldsymbol{w}(\boldsymbol{v}(\boldsymbol{y}))$ and $\boldsymbol{x} = \boldsymbol{v}(\boldsymbol{w}(\boldsymbol{x}))$ for any state on the SSM. These transformation functions can be defined as

$$\boldsymbol{v}(\boldsymbol{y}) = \boldsymbol{V}^\top \boldsymbol{y}, \tag{3.13}$$

$$\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{W}_0 \boldsymbol{x} + \boldsymbol{W} \boldsymbol{m}_{2:n_\mathrm{w}}(\boldsymbol{x}) \tag{3.14}$$

where the columns of $\boldsymbol{V} \in \mathbb{R}^{p \times n}$ span the spectral subspace of E and $\boldsymbol{W}_0 \in \mathbb{R}^{p \times n}$ and $\boldsymbol{W} \in \mathbb{R}^{p \times N_\mathrm{w}}$ are coefficient matrices. The vector $\boldsymbol{m}_{2:n_\mathrm{w}}(\boldsymbol{x}) \in \mathbb{R}^{N_\mathrm{w}}$ collects the evaluations at $\boldsymbol{x}$ of the family of all monomials of order 2 to n_w, where n_w is the desired order of the Taylor series approximating the SSM and N_w is the number of monomials of order 2 to n_w. The number N_w of monomials of order 2 to n_w can be calculated as $N_\mathrm{w} = \binom{n}{n_\mathrm{w}} - n$.

The autonomous dynamics on the SSM $\mathcal{W}(E)$ can be represented in an analogous way by the polynomial

$$\dot{\boldsymbol{x}}_\mathrm{aut} = \boldsymbol{r}_\mathrm{aut}(\boldsymbol{x}) := \boldsymbol{R_0} \boldsymbol{x} + \boldsymbol{R} \boldsymbol{m}_{2:n_\mathrm{r}}(\boldsymbol{x}) \tag{3.15}$$

where $\boldsymbol{R}_0 \in \mathbb{R}^{n \times n}$ and $\boldsymbol{R} \in \mathbb{R}^{n \times N_\mathrm{r}}$ are coefficient matrices. The vector $\boldsymbol{m}_{2:n_\mathrm{r}}(\boldsymbol{x}) \in \mathbb{R}^{N_\mathrm{r}}$ collects the evaluations at $\boldsymbol{x}$ of the family of all monomials of order 2 to n_r, where n_r is the desired order of the Taylor series approximating the autonomous dynamics on the SSM $\mathcal{W}(E)$ and N_r is the number of monomials of order 2 to n_r which is calculated as $N_\mathrm{r} = \binom{n}{n_\mathrm{r}} - n$.

As a first step of system identification, the matrix $\boldsymbol{V} \in \mathbb{R}^{p \times n}$ is computed by finding the n dominant modes of the system. This is done by applying a principal component analysis (PCA) [Pearson01] on $\boldsymbol{\mathcal{Y}}$ and selecting the n dominant modes. Note that for nonlinear systems the PCA is only able to obtain an estimate for the spectral subspace E, which, however, usually is sufficiently good for practical applications as long as the influence of the nonlinearities is sufficiently small. Now $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{p \times N_\mathrm{samples}}$ can be projected onto the reduced coordinates resulting in the autonomous reduced state matrix $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n \times N_\mathrm{samples}}$ which is calculated as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{V}^\top \boldsymbol{\mathcal{Y}}. \tag{3.16}$$

The autonomous reduced state matrix $\boldsymbol{\mathcal{X}}$ can also be written as

$$\boldsymbol{\mathcal{X}} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_{N_\mathrm{samples}} \end{bmatrix}. \tag{3.17}$$

In the next step, polynomial regression can be used to determine the coefficient matrices $\boldsymbol{W}_0 \in \mathbb{R}^{p \times n}$, $\boldsymbol{W} \in \mathbb{R}^{p \times N_\mathrm{w}}$, $\boldsymbol{R}_0 \in \mathbb{R}^{n \times n}$ and $\boldsymbol{R} \in \mathbb{R}^{n \times N_\mathrm{r}}$ by solving

$$(\boldsymbol{W}_0, \boldsymbol{W}) = \underset{\boldsymbol{W}_0, \boldsymbol{W}}{\operatorname{argmin}} \left|\left| \boldsymbol{\mathcal{Y}} - \boldsymbol{W}_0 \boldsymbol{\mathcal{X}} - \boldsymbol{W} \boldsymbol{\mathcal{M}}_{2:n_\mathrm{w}}(\boldsymbol{\mathcal{X}}) \right|\right|_F^2, \tag{3.18}$$

$$(\boldsymbol{R}_0, \boldsymbol{R}) = \underset{\boldsymbol{R}_0, \boldsymbol{R}}{\operatorname{argmin}} \left|\left| \dot{\boldsymbol{\mathcal{X}}} - \boldsymbol{R}_0 \boldsymbol{\mathcal{X}} - \boldsymbol{R} \boldsymbol{\mathcal{M}}_{2:n_\mathrm{r}}(\boldsymbol{\mathcal{X}}) \right|\right|_F^2. \tag{3.19}$$

Here the matrix $\boldsymbol{\mathcal{M}}_{2:n_\mathrm{w}}(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{N_\mathrm{w} \times N_\mathrm{samples}}$ collects the evaluations at $\boldsymbol{\mathcal{X}}$ of the family of all monomials of order 2 to n_w and is defined as

$$\boldsymbol{\mathcal{M}}_{2:n_\mathrm{w}}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \boldsymbol{m}_{2:n_\mathrm{w}}(\boldsymbol{x}_1) & \boldsymbol{m}_{2:n_\mathrm{w}}(\boldsymbol{x}_2) & \dots & \boldsymbol{m}_{2:n_\mathrm{w}}(\boldsymbol{x}_{N_\mathrm{samples}}) \end{bmatrix}. \tag{3.20}$$

The matrix $\boldsymbol{\mathcal{M}}_{2:n_\mathrm{r}}(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{N_\mathrm{w} \times N_\mathrm{samples}}$ is defined in an analogous way and collects the evaluations at $\boldsymbol{\mathcal{X}}$ of the family of all monomials of order 2 to n_r. The time derivative $\dot{\boldsymbol{\mathcal{X}}}$ can be numerically calculated using finite differences.

Learning the Control Matrix

Finally, the control matrix $\boldsymbol{B} \in \mathbb{R}^{n \times m}$ must be learned. For simplicity, and without loss of generality, it is assumed that $\epsilon = 1$. In a first step, a dataset with $N_\mathrm{samples,u}$ samples must be collected by applying a random, band-limited sequence of inputs $\boldsymbol{U} \in \mathbb{R}^{m \times N_\mathrm{samples,u}}$ to the system and recording the corresponding state trajectory $\boldsymbol{\mathcal{Y}}_\mathrm{u} \in \mathbb{R}^{p \times N_\mathrm{samples,u}}$. As the computation of the control matrix $\boldsymbol{B}$ relies on the evaluation of the learned autonomous dynamics $\boldsymbol{r}_\mathrm{aut}$, the inputs $\boldsymbol{U}$ should only attenuate dynamics that are also represented in the autonomous training set $\boldsymbol{\mathcal{Y}}$. The observed states are then projected onto the reduced coordinates, resulting in the reduced state matrix $\boldsymbol{\mathcal{X}}_\mathrm{u} \in \mathbb{R}^{n \times N_\mathrm{samples,u}}$ which is calculated as

$$\boldsymbol{\mathcal{X}}_\mathrm{u} = \boldsymbol{V}^\top \boldsymbol{\mathcal{Y}}_\mathrm{u}. \tag{3.21}$$

The control matrix $\boldsymbol{B}$ can then be calculated by solving

$$\boldsymbol{B} = \underset{\boldsymbol{B}}{\operatorname{argmin}} \left|\left| \dot{\boldsymbol{\mathcal{X}}}_\mathrm{u} - \boldsymbol{r}_\mathrm{aut}(\boldsymbol{\mathcal{X}}_\mathrm{u}) - \boldsymbol{B}\boldsymbol{U} \right|\right|_F^2, \tag{3.22}$$

where $\dot{\boldsymbol{\mathcal{X}}}_\mathrm{u}$ can be calculated using finite differences.

3.3.3 Application to Test System

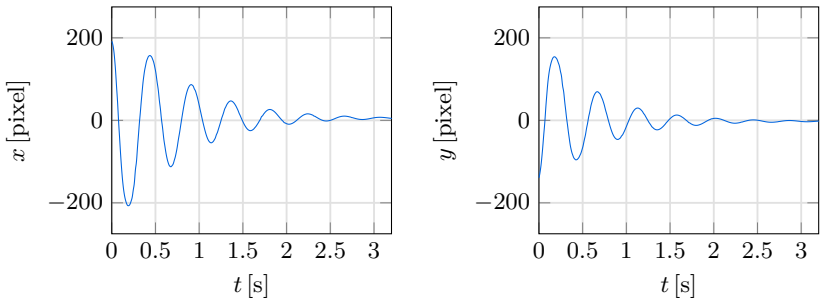
In the following, the spectral submanifold reduction (SSMR) algorithm is applied to the test system introduced in Sec. 3.1. The version of the test system with the "DS20" servos and the `Optitrack` 2D camera tracking system is used. The input of the system is the tendon length vector $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}$ and the output $\boldsymbol{z}$ is the x- and y-coordinate of the tip in pixel coordinates. In the following, the implementation of the SSMR algorithm in the MATLAB packages "SSMLearn" presented in [CenedeseEtAl22] and "SSMR-for-control" presented in [AloraEtAl23] is used. First, the training data collection is described. Then, the influence of the hyperparameters of the SSMR model is investigated. Finally, the performance of the SSMR model is examined in terms of accuracy and computational efficiency.

Training Data Collection

Learning the SSMR model requires a set of autonomous state trajectories $\boldsymbol{\mathcal{Y}}$ and a second set of non-autonomous state trajectories $\boldsymbol{\mathcal{Y}}_{\mathrm{u}}$. In a first step the autonomous trajectories are recorded. To record the autonomous trajectories, the system input is set to $\boldsymbol{s} = \mathbf{0}$, which corresponds to the straight upright configuration in the quasi-static case. The easiest way to record autonomous trajectories is to simply apply an initial deflection to the soft robot, e.g. by hand, and record the resulting decaying trajectory. An example of such a trajectory is shown in Fig. 3.11. However, with this method the achievable speed at the edges of the workspace is low, limiting the ability to capture the dynamics of the system in these areas.

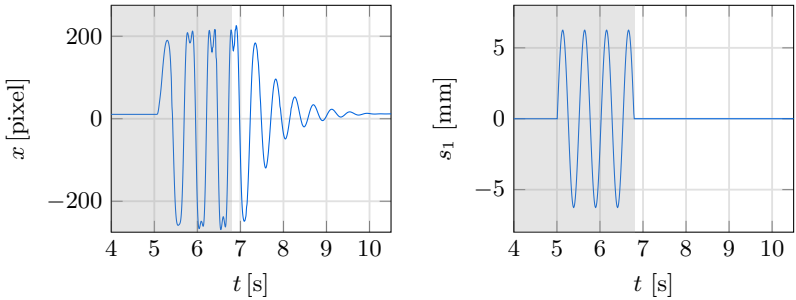
An alternative is to perform an oscillation excitation maneuver on the robot to achieve an initial deflection and velocity of the system. This is done by applying a sinusoidal motion of approximately the soft robots eigenfrequency to the system using the feedforward inverse kinematic controller presented in Sec. 5.1, which is based on the kinematic model presented in Sec. 3.2. Note that while the kinematic controller does not take into account the dynamics of the system, it is sufficient to generate input signals that lead to strong oscillations of the system. In Fig. 3.12 the oscillations in x-direction of the system and the system input s_1 for the first servo are shown. The oscillations in y-direction and the system inputs for the other two servos are qualitatively comparable and are therefore omitted here. In both diagrams, the non-autonomous phase is highlighted in gray and is not part of the autonomous trajectory, since the system inputs are active. This approach allows for slightly stronger dynamics at the edges of the workspace than by simply applying an initial deflection.

Autonomous trajectories with even larger dynamics at the edges of the workspace are difficult to capture, because the required initial conditions easily lead to



a) x-coordinate. **b)** y-coordinate.

Figure 3.11: Recording of an autonomous trajectory starting from an initial deflection.

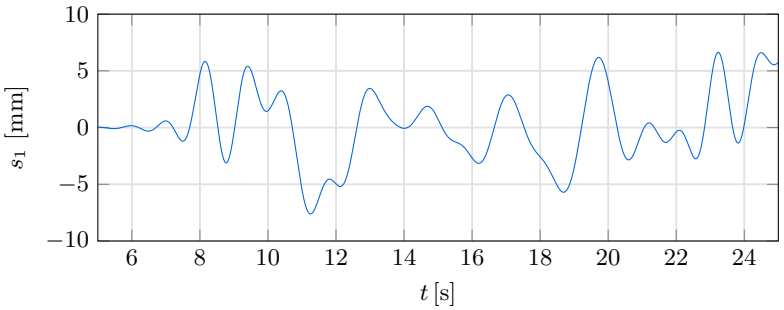


a) x-coordinate of the tip position. **b)** Input tendon length s_1.

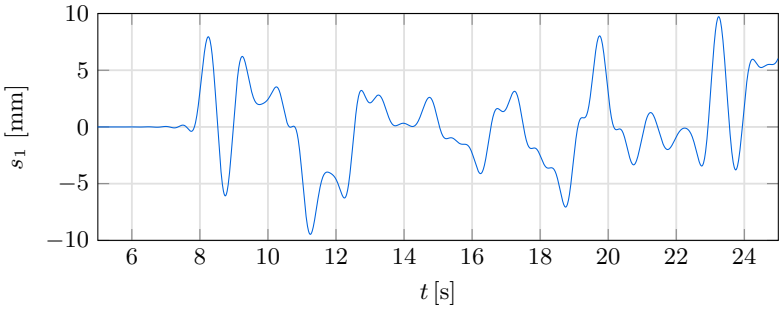
Figure 3.12: Recording of an autonomous trajectory with an initial deflection and a initial velocity resulting from a maneuver to excite oscillations. The grey area marks the time when the servo motors are active to excite oscillations.

the robot leaving its workspace and getting damaged. With non-autonomous trajectories, such dynamic behavior at the edges of the workspace can be achieved because appropriate system inputs can be applied to prevent the robot from leaving the workspace. However, non-autonomous trajectories are not suitable for learning the dynamics of the system with the SSM approach. The training data set consists of 70 autonomous trajectories, each with a duration of approximately 3.5 seconds and a sample rate of 100 Hz. Two-thirds of these trajectories are recorded with an initial deflection only, and the remaining third are recorded with an initial deflection and velocity.

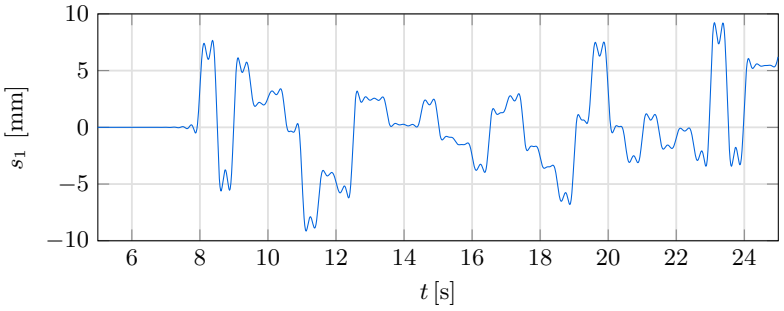
In a second step, the non-autonomous trajectories are recorded. For this, several random sets of system inputs $\boldsymbol{s}$ are sampled with sampling times of $0.2\,\mathrm{s} \ldots 1\,\mathrm{s}$. The redundancy of the actuation is resolved as described in Sec. 3.1.2 for the quasi-static NN model by restricting the inputs $\boldsymbol{s}$ so that the sum of the tendon forces $\hat{F}$ is constant in the quasi-static case. Then, Butterworth low-pass filters of order 8 with cut-off frequencies of $1\,\mathrm{Hz} \ldots 4\,\mathrm{Hz}$ are applied to the sets of system inputs. The result is the removal of high frequency components, which are not represented in the autonomous training set, from the input signal while preserving a wide range of lower frequencies. These sets of system inputs are then collected in the system input matrix $\boldsymbol{U}$, applied to the soft robot, and the tip position measurements are collected in the matrix $\boldsymbol{\mathcal{Y}}_\mathrm{u}$. An excerpt of the generated system inputs is shown in Fig. 3.13. In total, 28 min of non-autonomous trajectories sampled with a sample rate of 100 Hz are used for training.



a) Cut-off frequency 1 Hz.



b) Cut-off frequency 2 Hz.



c) Cut-off frequency 4 Hz.

Figure 3.13: Example of a random input sequence sampled at 2 Hz and filtered with low pass filters with three different cut-off frequencies.

Hyperparameters of the SSMR Model

Several hyperparameters must be selected to derive the SSMR model. Most important are the dimensionality n of the reduced subspace E, the order n_w

of the polynomial $\boldsymbol{w}(\boldsymbol{x})$ describing the SSM, and the order n_r of the polynomial $\boldsymbol{r}_\mathrm{aut}(\boldsymbol{x})$, which approximates the reduced dynamics on the submanifold. In addition, the dimension p of the measured state $\boldsymbol{y}$, which is set by the number of time embeddings affects the model performance. The hyperparameters are chosen by testing different combinations within reasonable ranges after some initial experiments. For each evaluation, the model configured with the selected hyperparameter combination is used to predict 20 additionally recorded non-autonomous trajectories based on random input sequences with a duration of 25 s each. The accuracy of these predictions is then analyzed by calculating the root mean squared error e_RMS defined as

$$e_\mathrm{RMS} = \sqrt{\frac{1}{N_\mathrm{samp}} \sum_{i=1}^{N_\mathrm{samp}} ||\boldsymbol{z}_{\mathrm{mod},i} - \boldsymbol{z}_{\mathrm{meas},i}||^2} \tag{3.23}$$

between the simulated trajectories $\boldsymbol{z}_{\mathrm{mod},i}$ and the corresponding tip position measurement $\boldsymbol{z}_{\mathrm{meas},i}$ where N_samp is the total number of samples within the recorded trajectories.

The order n_w of the polynomial $\boldsymbol{w}(\boldsymbol{x})$ is tested in the range $n_\mathrm{w} = 1 \dots 4$ and the dimension p of the measured state $\boldsymbol{y}$ is tested in the range $p = 10 \dots 18$. The order n_r of the polynomial $\boldsymbol{r}_\mathrm{aut}(\boldsymbol{x})$ is fixed at $n_\mathrm{r} = 1$ and the dimension n of the subspace E is fixed at $n = 4$, since all other settings lead to higher errors regardless of the other parameters. The results of the experiments are shown in Fig. 3.14. It can be observed that the polynomial order n_w of the polynomial $\boldsymbol{w}(\boldsymbol{x})$ has a large impact on the error, and especially orders greater than three cause the largest increase in error. Reducing the dimension p also reduces the error, but the effect is smaller.

Based on this, the final hyperparameters of the model are chosen as listed in Tab. 3.1. This results in linear reduced dynamics on the submanifold, allowing many established linear techniques to be used to analyze and control the otherwise nonlinear system.

parameter	**symbol**	**value**
Dimension of the reduced subspace E	n	4
Order of the polynomial $\boldsymbol{r}_\mathrm{aut}(\boldsymbol{x})$	n_r	1
Order of the polynomial $\boldsymbol{w}(\boldsymbol{x})$	n_w	3
Dimension of the measured state vector $\boldsymbol{y}$	p	10

Table 3.1: Overview of the chosen hyperparameters of the SSMR model.

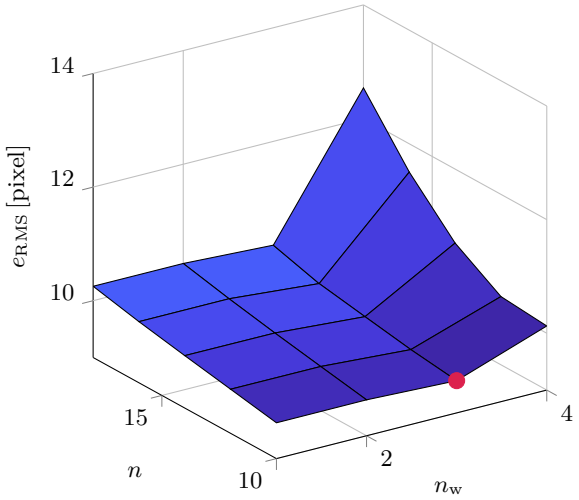


Figure 3.14: Influence of the order n_{w} of the polynomial $\boldsymbol{w}(\boldsymbol{x})$ and the dimension n of the reduced subspace on the model accuracy. The best hyperparameter combination is highlighted with a red dot.

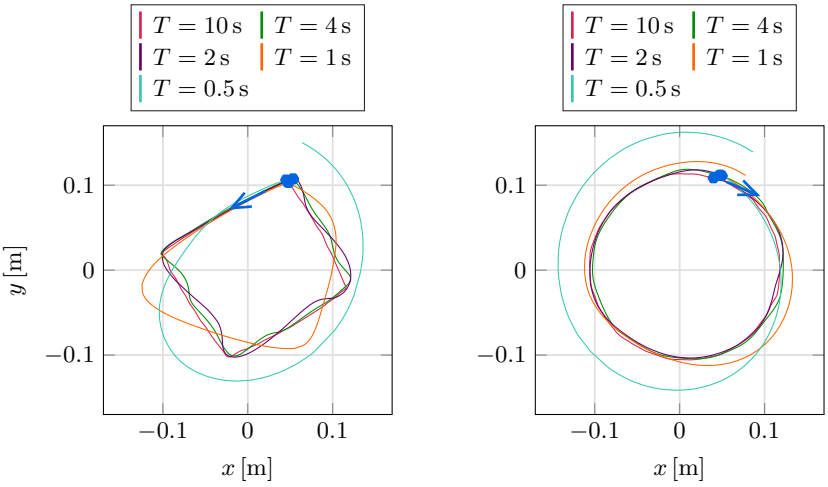
Model Performance

In order to evaluate the accuracy of the derived SSMR model, different system inputs are applied to both, the physical system and the simulation model. The system inputs are generated using the inverse quasi-static feedforward controller which is presented in Sec. 5.1. This controller is based on the data-driven quasi-static forward model presented in Sec. 3.2. The system inputs are generated assuming a quasi-static behavior of the soft robot. System inputs are generated for tracking a rectangular trajectory of size 140 mm × 160 mm and a circular trajectory of radius 110 mm with constant velocity. The initial velocity is set to zero. The generated system inputs are then time scaled so that the trajectories are tracked with a period of $T = 10\,\mathrm{s}$, 4 s, 2 s, 1 s, and 0.5 s. Note that since quasi-static behavior is assumed for trajectory generation, it cannot be expected that the trajectories with high velocity are tracked accurately. However, a comparison between the behavior of the SSMR model and the physical system is possible. These system inputs are then applied to the physical robot and the SSMR model. The simulation of the SSMR model is performed using the MATLAB solver `ODE15s`. For simulation, only the initial conditions and the system inputs are applied. The simulation is then performed open-loop over the whole period time T.

In Fig. 3.15 the recorded trajectories of the physical robot are shown for the rectangular and circular trajectories and different values for the period time T. These form the basis for the evaluation of the SSMR model. The blue arrows

indicate the direction of the trajectories. It can be seen that for a large period time of $T = 10\,\mathrm{s}$, corresponding to almost quasi-static behavior of the soft robot, the recorded trajectory deviates only slightly from the specified trajectory. However, for lower period times T it can be observed that the recorded trajectories increasingly deviate from the specified shape. This behavior is expected and is due to the dynamic behavior of the soft robot, which was not taken into account when generating the inputs. It can be seen that the starting points of the different trajectories vary slightly. This can be explained by the slight hysteretic behavior of the physical soft robot.

In Figs. 3.16 and 3.17 the recorded trajectories of the physical robot and the simulation results are shown for the rectangular and circular trajectories and different values for the period time T. The quasi-static trajectories with a period time of $T = 10\,\mathrm{s}$ are simulated accurately with a mean error of $17.75\,\mathrm{pixel}\,(\approx 8\,\mathrm{mm})$. Reasonably accurate simulation results with a mean error of $45.98\,\mathrm{pixel}(\approx 24\,\mathrm{mm})$ can still be achieved for trajectories with moderate period times of $T = 4\,\mathrm{s}$ and $T = 2\,\mathrm{s}$. For increasingly smaller period times, i.e. higher velocities, and thus stronger dynamics, the simulation model increasingly overestimates the dynamics, which leads to larger simulation errors. For trajectories with a period time of $T = 0.5\,\mathrm{s}$ this effect results in a very large mean simulation error of 159.46 pixel. This is mainly due to large velocities occurring at the edge of the working space,

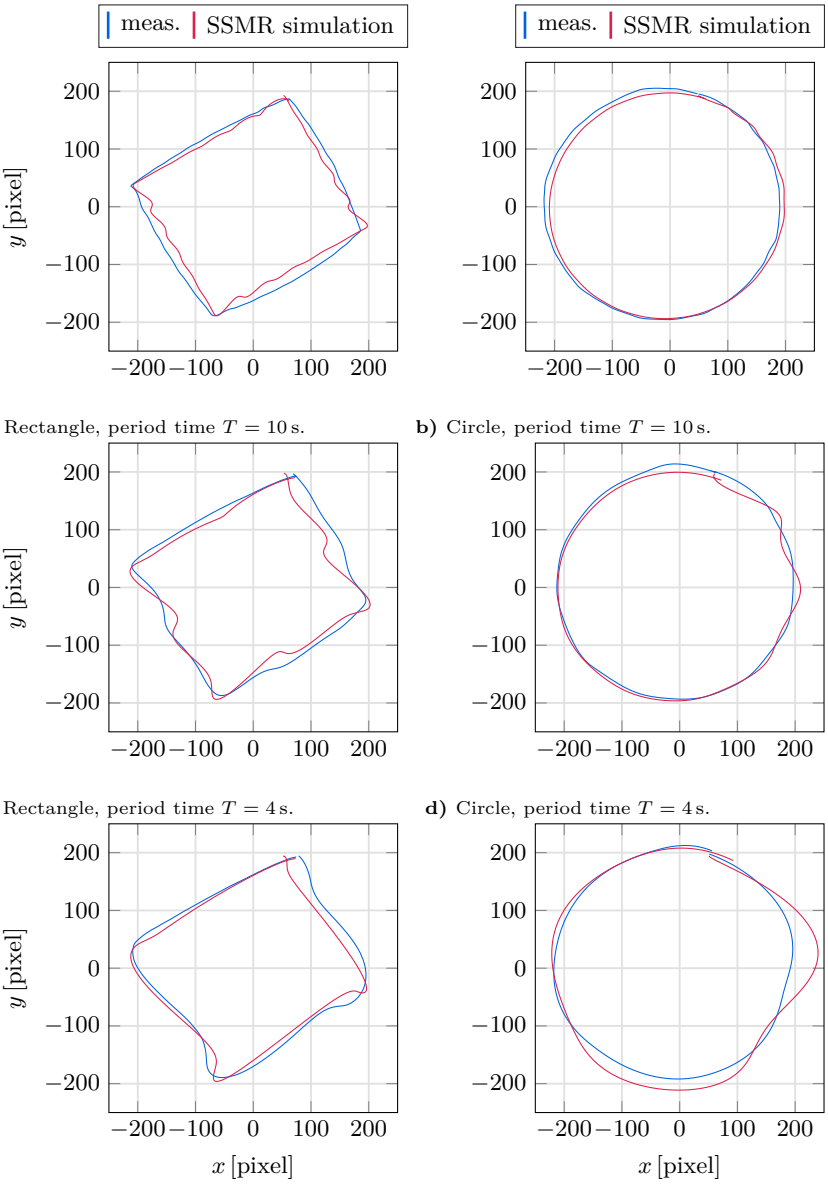


a) Rectangular trajectory.

b) Circular trajectory.

Figure 3.15: Measurements of circular and rectangular trajectories in experiment using system inputs generated with a inverse quasi-static model at five different period times T each.

which are not represented in the training set for the system dynamics and require the model to extrapolate. This can be seen in Fig. 3.18, which shows the velocity profile of an exemplary autonomous trajectory from the training set and the velocity profile for the rectangular trajectory with $T = 0.5\,\mathrm{s}$.



a) Rectangle, period time $T = 10\,\mathrm{s}$.

b) Circle, period time $T = 10\,\mathrm{s}$.

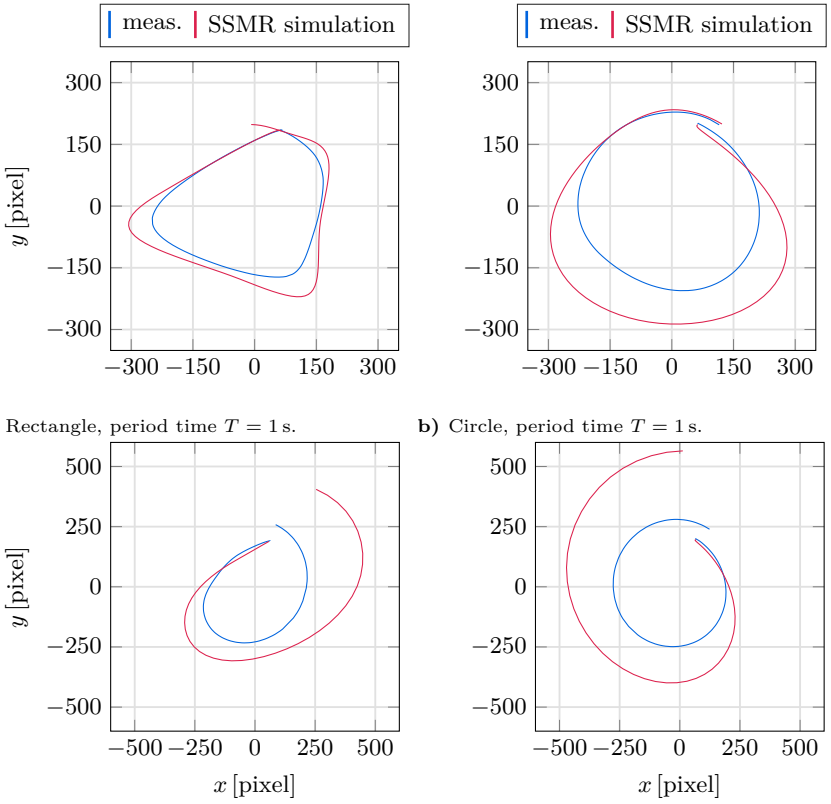
c) Rectangle, period time $T = 4\,\mathrm{s}$.

d) Circle, period time $T = 4\,\mathrm{s}$.

e) Rectangle, period time $T = 2\,\mathrm{s}$.

f) Circle, period time $T = 2\,\mathrm{s}$.

Figure 3.16: Measurements and the corresponding simulation results of the SSMR model of the rectangular and circular trajectories for the different durations $T = 10\,\mathrm{s}$, $4\,\mathrm{s}$, $2\,\mathrm{s}$.



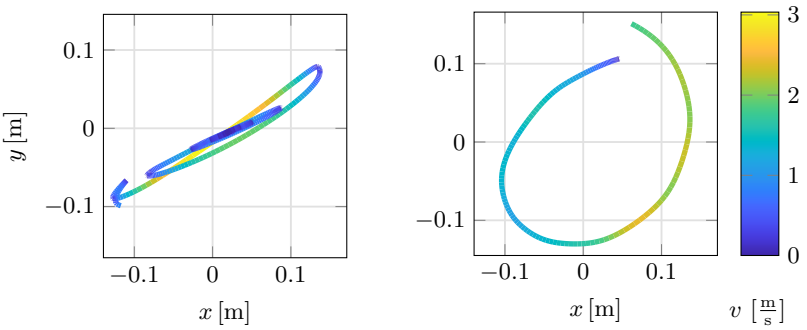
a) Rectangle, period time $T = 1\,\mathrm{s}$.

b) Circle, period time $T = 1\,\mathrm{s}$.

c) Rectangle, period time $T = 0.5\,\mathrm{s}$.

d) Circle, period time $T = 0.5\,\mathrm{s}$.

Figure 3.17: Measurements and the corresponding SSMR simulations of the rectangular and circular trajectories for the different durations $T = 1\,\mathrm{s},\ 0.5\,\mathrm{s}$.



a) Autonomous training trajectory. **b)** Rectangle, period time $T = 0.5\,\mathrm{s}$.

Figure 3.18: Velocity profiles of an autonomous training trajectory and the measurement of the $T = 0.5\,\mathrm{s}$ duration rectangle trajectory in comparison.

Sensors

As soft robotic applications become more complex, control requirements increase. However, accurate knowledge of the current position and orientation of the soft components is essential for reliable feedback trajectory tracking control and positioning of soft robots. Another application for soft sensors is soft grippers, which are currently one of the most popular applications of soft robots [LeeEtAl17a]. While the soft structure of soft grippers allows them to adapt to the shape of the gripped objects even without sensory feedback, this is not sufficient for increasingly demanding applications. Here, the integration of sensors into the soft robot allows the gripping process to be monitored and controlled in order to achieve a uniform and sufficiently firm grip and an overall more reliable gripping process.

In soft robotics research, external camera tracking systems are widely used to provide sensory feedback on the deformation of soft robots. For real-world applications, however, internal sensors are essential to enable use in different environments with different environmental conditions. However, these sensors must not compromise the softness of the soft robot. For this reason, conventional rigid sensors usually cannot be used in soft robotics. Also, most flexible conventional sensors, such as strain gauges, cannot be used because they cannot withstand the large strains that occur in soft robots. Therefore, various soft sensors such as resistive [CianchettiEtAl12, LiEtAl17, PolygerinosEtAl17, TanEtAl22], capacitive [RobertsEtAl13, YaoZhu14, LiEtAl17], magnetic [JentoftHowe11, OzelEtAl16], or optical [YiEtAl10, PolygerinosEtAl11, RyuEtAl12, AbayazidEtAl13, WangEtAl16, GeEtAl16a, GallowayEtAl19], sensors are currently being developed for soft robots.

In soft robotics, resistive and optical sensors are the most widely used sensors that can be integrated into robots. Existing resistive sensors, such as [TanEtAl22], usually have strong hysteretic behavior, which makes their evaluation and application to real systems very difficult. Additionally, they are usually sensitive to electrical noise. Advantages of optical sensors are their immunity to electrical noise and high flexibility [LiEtAl17]. Existing soft optical sensors, such as [AbayazidEtAl13, WangEtAl16, GeEtAl16a, GallowayEtAl19], are mostly based on fiber Bragg gratings and are usually very expensive and depend on large, rigid evaluation components. Therefore, they usually cannot be

fully integrated into soft robots. Other soft optical sensors such as presented in [RyuEtAl12], are difficult to manufacture and not commercially available. This currently limits the applicability of existing soft optical sensors.

For soft robots, bending is usually the dominant deformation and therefore an estimation of the curvature is necessary. Also one of the most popular modeling techniques for soft robots, the piecewise constant curvature (PCC) model, see Sec. 2.1, is based on a description of bending. The model divides a soft robot into pieces of constant curvature. This makes curvature sensors especially interesting for applications using controllers based on the PCC model.

In this chapter, two low-cost, loss-based fiber-optic curvature sensors are presented that can be fully integrated into soft robots. First, a planar curvature sensor is presented in Sec. 4.1. The curvature sensor can determine the magnitude of the curvature in two spatial directions, but not the direction, which limits its use mainly to planar applications. Based on this sensor, a spatial curvature sensor is presented in Sec. 4.2 that can determine both the magnitude and the direction of curvature in two spatial directions. For both sensors, the design and fabrication are described, followed by an evaluation of the sensor accuracy and an experimental investigation of the application to soft robots. Large parts of this chapter have also been published in [GrubeSeifried22a, GrubeSeifried23, GrubeEtAl25].

4.1 Planar Curvature Sensor

As a first sensor, a low-cost fiber-optic and easy to fabricate curvature sensor for mainly planar applications is presented. This sensor can determine the magnitude of curvature in two spatial directions, but not the direction. First, the design and fabrication of the sensor is discussed, followed by the experimental evaluation of the accuracy of the sensor. Finally, the application of the curvature sensor to a soft robotic gripper for shape estimation is experimentally demonstrated.

4.1.1 Sensor Design and Fabrication

The curvature sensor consists of short segments of polymer optical fiber (POF) surrounded by a flexible tube to keep them aligned. To achieve high accuracy and sensitivity of the sensor, the optical fiber should be stiffer than the surrounding silicone tube and must fit tightly into the tube. This allows for a sufficiently

large angle between the optical axis of successive fiber segments when bending the curvature sensor. One end of the tube is connected to an infrared light emitting diode (IR LED) and the other end to a phototransistor. The sensor is shown in Fig. 4.1. The measurement circuit of this sensor consists only of a small analog-to-digital converter (ADC) with a size of $\approx 5 \times 5\,\text{mm}$ and a few resistors. Its small size allows it to be fully integrated into soft robots without compromising overall softness. The sensor can determine the magnitude of the curvature in two spatial directions, but not the direction of the curvature. Therefore, the following presentation is limited to the planar motion of a soft robot.

Measurement principle

The core component of this sensor is a polymer optical fiber (POF). POFs consist of a core surrounded by a cladding material with a lower refractive index. Light entering the POF at an angle less than the acceptance angle θ_{acc} of the POF is retained in the core of the fiber by total reflection between cladding and core. The acceptance angle depends on the refractive indices of the core n_{core}, the cladding n_{cladding} and the surrounding medium n_0. The working principle of this sensor is based on the optical power loss between successive segments of POF. The LED provides a constant radiant flux $\Phi_{\text{e,LED}}$. At each boundary between successive segments of POF a certain percentage of light is lost, depending on the angle α_i, see Fig. 4.1, between the POF segments. The larger the angle α_i, the less light is transmitted to the next POF segment [Hui09, chap. 1]. Note that the percentages of the remaining light are multiplied along the sensor and not summed. Finally, with the phototransistor the radiant flux $\Phi_{\text{e,pt}}$ that leaves the last POF segment is measured. This measurement of the radiant flux $\Phi_{\text{e,pt}}$ allows conclusions to be drawn about the deformation of the sensor. In general, for a sensor with N POF segments, the radiant flux $\Phi_{\text{e,pt}}$ of the light reaching

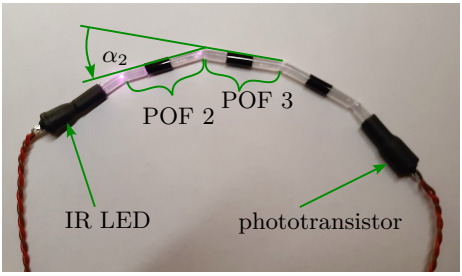


Figure 4.1: Sensor in bend configuration. Exemplary the angle α_2 between segment 2 and segment 3 is shown.

the last POF segment and thus the phototransistor can be described by

$$\Phi_{\mathrm{e,pt}} = \Phi_{\mathrm{e,LED}} \underbrace{\prod_{i=1}^{N-1} f(\alpha_i)}_{\tilde{f}(\alpha_1,\ldots,\alpha_{N-1})} , \tag{4.1}$$

see [Hui09, chap. 1]. The nonlinear function f describes the behavior of the sensor. It depends on the angle α_i between the POF segments of the sensor, see Fig. 4.1. This function is bijective and monotonically decreasing in the operating range of the sensor, therefore its inverse function f^{-1} exists. Both, the function f and its inverse f^{-1} usually have to be determined experimentally. The inverse of the function $\tilde{f}(\alpha_1, \ldots, \alpha_{N-1})$, however is not unique for more than $N = 1$ segments, because there are infinitely many combinations of angles α_i that lead to the same light loss. For practical applications, it is often not the individual angles α_i between the successive POF segments of the sensor that are important, but the total angle α over the sensor, which is defined as sum over all individual angles α_i between all POF segments. The total angle α can be calculated as

$$\alpha = \sum_{i=1}^{N-1} \alpha_i . \tag{4.2}$$

In the following, the mapping between the phototransistor current and the total angle α is determined directly. Obviously, this mapping is not unique, but still depends on the individual angles α_i. However, it is shown in Sec. 4.1.2 that as long as the angles α_i are not too different, the introduced error is small.

The radiant flux $\Phi_{\mathrm{e,LED}}$ of the IR LED can be easily kept constant by applying a constant electrical current to the IR LED. The radiant flux $\Phi_{\mathrm{e,pt}}$ can be measured with the phototransistor and an analog-to-digital converter (ADC). The electrical current flow through the phototransistor depends linearly on the radiant flux $\Phi_{\mathrm{e,pt}}$ as long as the phototransistor is not saturated.

Measurement circuit

The measurement circuit used consists of a light emitting part and a light receiving part, which are galvanically isolated and only optically connected by the POF. The measurement circuit is shown in Fig. 4.2. The main components are an IR LED of type “SFH 4555”, a phototransistor of type “SFH 313 FA”, and a 12-bit analog-to-digital converter (ADC) of type “MCP3208”. These components are not critical, and other components with similar characteristics, especially smaller LEDs and phototransistors, can also be used. The ADC has an SPI interface, which allows easy communication with many microcontrollers or single board computers like the Raspberry Pi. The series resistor R_1 of the IR LED has to be chosen in a way that the maximum current of the IR LED is not exceeded. If the

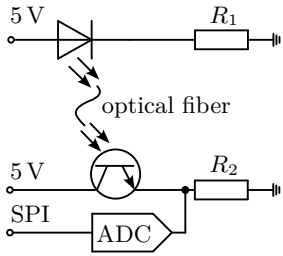


Figure 4.2: Measurement circuit.

maximum brightness of the IR LED is not needed, lower values for the resistor are possible. The ADC measures the voltage drop across the resistor R_2, which is proportional to the current flowing through the phototransistor. As a rule of thumb, the resistor R_2 should be chosen to measure about 90 % of the maximum phototransistor current when the sensor is in a straight-line configuration to make the best use of the ADC's resolution and do not drive the phototransistor into saturation. Too large values for the resistor R_2 lead to saturation of the phototransistor, so that small angles can no longer be distinguished. Smaller values for the resistor R_2 reduce the effective resolution of the sensor.

Fabrication

For fabrication, in step 1 the POF has to be cut into segments. Here, POF with diameters of 1 mm and 2 mm are used. Thinner fibers can also be used, but the fabrication becomes more difficult, and it is likely that the light loss at the coupling points between the fiber and the IR LED or phototransistor increases, requiring more powerful LEDs to compensate. In step 2, the ends of the fiber must be ground into a semicircular shape to allow easy bending of the sensor without changing its length. Polishing the ends is not necessary, but results in less unwanted light loss, so lower wattage IR LEDs can be used. In step 3, one POF segment is attached to the LED and another to the phototransistor. 3D printed connectors are used for proper alignment. It is important that there is no air gap or opaque adhesive between the fiber and the IR LED or phototransistor to minimize unwanted light loss. In a fourth step, the prepared fiber segments are placed in a flexible tube to align them properly. The fiber segment with the IR LED is at one end and the phototransistor is at the other end. Since infrared light is hardly present in the environment, transparent tubes can be used. However, depending on the lighting conditions, colored tubes may slightly reduce distortions due to ambient light and are therefore recommended. In a first version of this sensor, silicone tubes are used to connect the fiber segment because of their high flexibility. In addition, PU (polyurethane) tubes are used in between to allow the sensor to be easily glued to the soft robot. However,

in updated versions of this sensor, the silicone tube is directly glued to the soft robot or even directly integrated into the silicone matrix of a soft robot. The components of the sensor are shown in Fig. 4.3. The fully assembled sensor in bend configuration is shown in Fig. 4.1.

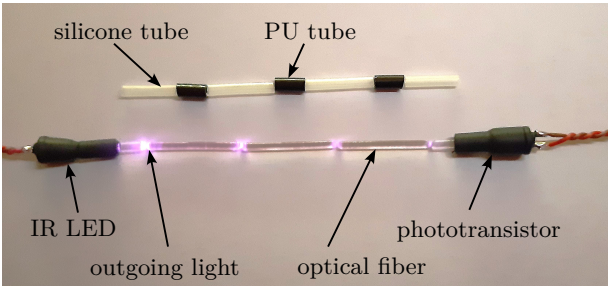


Figure 4.3: Components of the sensor.

Kinematics

For the soft curvature sensor introduced here, the angles α_i between successive segments are the most natural way to describe the planar deformation of the sensor. For soft robots, however, the PCC model is often used, which uses the curvatures κ_i of multiple segments to describe the deformation, see Sec. 2.1. In Fig. 4.4 the kinematics of the curvature sensor and the kinematics of a 2D PCC model are visualized. In the following, the relationship between the angles α_i from the sensor and the curvature κ_i of a PCC model is shown.

For the application of the curvature sensor to soft robots a relation between the two ways of describing the deformation has to be found. In a first step, the local bending angle $\tilde{\alpha}_i$ of each segment is introduced, see Fig. 4.4. With the definition of the curvature κ_i,

$$\kappa_i = \frac{\tilde{\alpha}_i}{\ell_i} \tag{4.3}$$

and the geometric relationship between two successive bending angles

$$\tilde{\alpha}_i = 2\alpha_i - \tilde{\alpha}_{i-1}. \tag{4.4}$$

The curvature of one PCC secment can be determined recursively from the angles α_i. This results in

$$\kappa_i = \frac{1}{\ell_i}\left(\alpha_i + \kappa_{i-1}\ell_{i-1}\right). \tag{4.5}$$

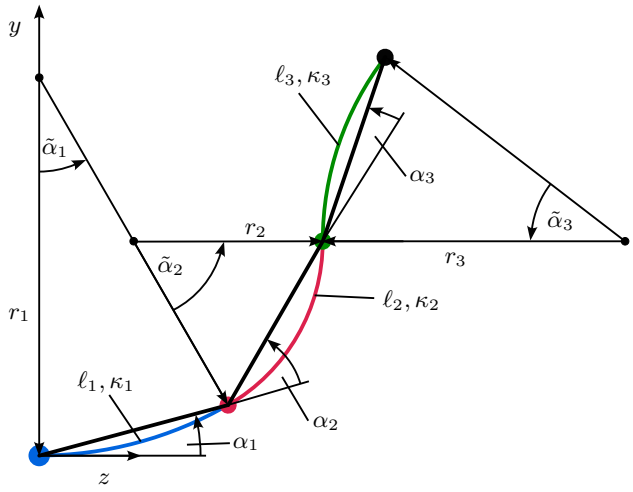


Figure 4.4: Kinematics of the curvature sensor and a planar PCC model.

This relationship can also be inverted to obtain the angle α_i from the curvature κ_i by

$$\alpha_i = \frac{1}{2}\left(\kappa_i \ell_i - \kappa_{i-1}\ell_{i-1}\right). \tag{4.6}$$

4.1.2 Experimental Evaluation of the Sensor Performance

To evaluate the performance of the optical bending sensor in terms of accuracy and reproducibility, a simple version of the sensor with only three POF segments is fabricated. The first and last POF segments are each actuated by a servo, and the middle POF segment is clamped. The test setup is shown in Fig. 4.5. With the servos the angles α_1 and α_2 between the first and the second segment and between the second and the third segment, respectively, can be set independently. In a first step the case of constant curvature is examined, where $\alpha_1 = \alpha_2 = \alpha/2$. In a second step, the behavior of the sensor is examined when the curvature is not constant. In the following, the measured currents of the phototransistor are normalized to a value range of 0 % for the lowest measured current and 100 % for the highest measured current and are denoted $i_{\mathrm{pt,rel}}$.

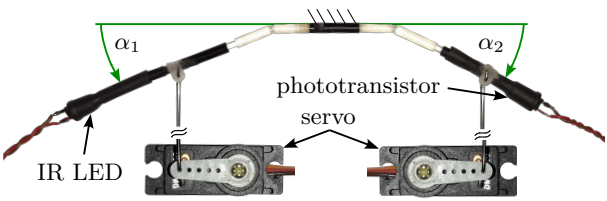


Figure 4.5: Three-segment curvature sensor for experimental evaluation of the sensor performance.

Constant Curvature

To evaluate the performance of the sensor at constant curvature, the angles α_1 and α_2 are varied between 0° and 30° with $\alpha_1 = \alpha_2$. This leads to bending with constant curvature and bending angles of $0° < \alpha < 60°$. This variation has a sinusoidal form with a period time of $T = 10\,\mathrm{s}$ and is run for 25 periods. Angles larger than 30° between successive segments are not considered because larger angles lead to reduced accuracy of the sensor and stronger nonlinearities in the sensor behavior. If a large total angle α, resulting in angles α_i of more than 30°, is required, a sensor with more but shorter segments should be used. Further investigation has shown that the choice of period Time T has no effect on the behavior of the sensor. In Fig. 4.6 a section of the measurement series is presented, showing the reference angle α_{ref} and the measured current $i_{\mathrm{pt,rel}}$.

To determine the total angle α from the measured phototransistor current, a function $\alpha(i_{\mathrm{pt,rel}})$ must be found that maps the measured phototransistor currents $i_{\mathrm{pt,rel}}$ to the corresponding total angle α. In the following, this function is approximated with polynomials $\alpha_{\mathrm{fit},k}(i_{\mathrm{pt,rel}})$ of different polynomial order k of the form

$$\alpha(i_{\mathrm{pt,rel}}) \approx \alpha_{\mathrm{fit},k}(i_{\mathrm{pt,rel}}) = \sum_{j=0}^{k} b_j i_{\mathrm{pt,rel}}^{j}. \tag{4.7}$$

The coefficients b_j of the polynomials are determined using polynomial regression. The polynomial regression is performed with the MATLAB function `polyfit`.

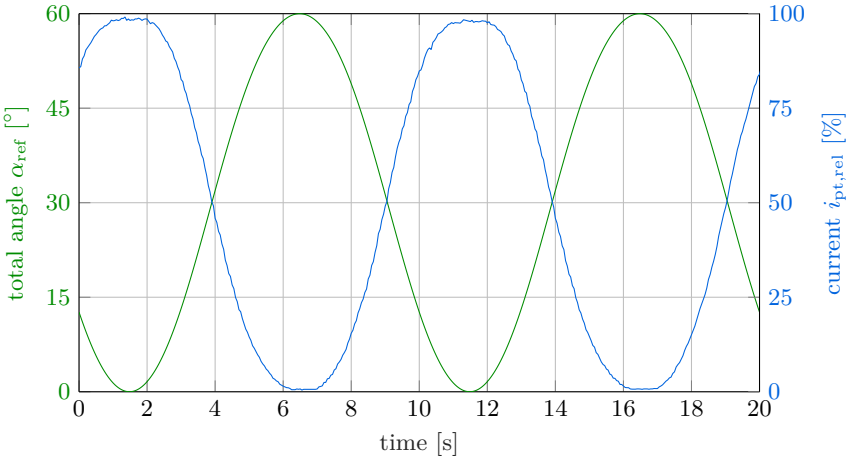


Figure 4.6: Section of measurement series.

For calibration, the measured data set recorded with a sampling rate of 50 Hz is divided into a training set and a test set. The training set contains the first 10 s of the measured data, which is a full period of the sinusoidal motion. Further investigation has shown that a larger training set does not improve the results. The test set contains the remaining data and is used to evaluate the accuracy of the sensor. The polynomial coefficients obtained are listed in Tab. 4.1.

To validate the sensor performance, the first step is to evaluate the accuracy of the calibration polynomials on the test set. As an error measure, the mean error on the test set with N samples is defined as

$$e = \frac{1}{N} \sum_{i=1}^{N} |\alpha_{\text{fit}} - \alpha_{\text{ref}}| \,. \tag{4.8}$$

Where α_{fit} is the angle obtained from equation (4.7) with the curvature sensor and α_{ref} is the reference value of the angle set by the servos.

In Fig. 4.7 the total angle α is plotted over the normalized measured phototransistor. Additionally, polynomial fits with polynomials of order 1, 3 and 5 are shown. It can be seen that for moderate angles in the range $10° \ldots 50°$ the relationship between phototransistor current and total angle is nearly linear. However, the polynomial fit with higher order polynomials is slightly better, especially for angles below 10° and above 50°. With a linear fit, a mean error of $e_1 = 1.19°$ with a standard deviation of 0.71° and a maximum error below 3.3° could be achieved. With a polynomial fit using a polynomial of degree 5, the mean error could be reduced to $e_5 = 0.34°$ with a standard deviation of 0.29° and a maximum error of 1.7°. Higher degree polynomials do not significantly reduce the error. Therefore, polynomials of degree 5 are used in the following.

In a second step, the measurement noise is analyzed. Therefore, in a second experiment, the sensor is held in a straight position while the ADC value is recorded over a period of 10 min. Here a standard deviation of the ADC value of 1.4 bit could be achieved. For the measurement of the total angle α this corresponds to a standard deviation of only 0.09°. This low noise level allows to easily determine the angular velocities $\dot{\alpha}$ by numerical differentiation of the

Table 4.1: Polynomial coefficients b_j for different polynomial orders k obtained by the calibration for constant curvature.

k	b_0	b_1	b_2	b_3	b_4	b_5
1	58.8	-0.571	-	-	-	-
2	58.5	-0.542	$-2.9 \cdot 10^{-4}$	-	-	-
3	60.1	-0.860	$8.45 \cdot 10^{-3}$	$-5.9 \cdot 10^{-5}$	-	-
4	59.8	-0.774	$3.93 \cdot 10^{-3}$	$1.48 \cdot 10^{-5}$	$-3.74 \cdot 10^{-7}$	-
5	60.2	-0.952	0.0187	$-4.08 \cdot 10^{-4}$	$4.54 \cdot 10^{-6}$	$-1.99 \cdot 10^{-8}$

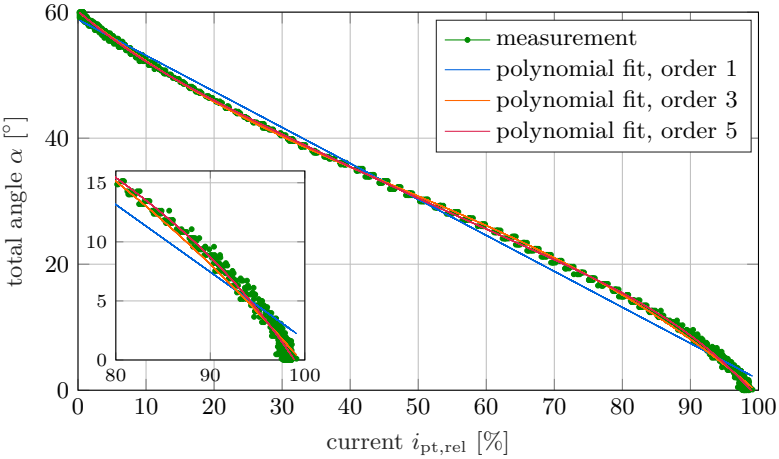


Figure 4.7: Total angle over measured phototransistor current.

bending angle α. The angular velocities are important for many control algorithms, such as those discussed in chapter 6.

Non-constant curvature

For practical applications, the performance of the sensor is also of interest when constant curvature cannot be guaranteed. As it is not possible to determine both angles α_1 and α_2 from the measured current i_{pt}, the total angle defined in equation (4.2) is considered again in the following. It can be seen from equation (4.1) that in general the total angle cannot be uniquely determined if the ratio between the angles α_i is unknown. However, it can be expected that a good approximation is possible if the angles are not too different.

To evaluate the performance of the sensor for non-constant curvature, the angles α_1 and α_2 are varied sinusoidally between $0°$ and $30°$. A period time of $T_1 = 7.1\,\mathrm{s}$ is used for the angle α_1 and a period time of $T_2 = 97.7\,\mathrm{s}$ is used for the angle α_2. In this way, several different combinations of the angles α_1 and α_2 are obtained. The total measurement time is 30 min. In Fig. 4.8 a section of the measurement series is shown. It shows the reference angle α_{ref} and the measured current $i_{\mathrm{pt,rel}}$.

Since it turns out that it is not possible to obtain good results for very different ratios of α_1/α_2, and the case of constant curvature is the most relevant case for practical applications, it makes sense to consider only bending with constant curvature during calibration. In general, the calibration obtained above for the constant curvature case can be reused. However, due to minor changes in the mechanical setup, the sensor is recalibrated here. Calibrating the sensor for non-constant curvature is similar to the constant curvature calibration described above. Only the division of the measurement series into a training set and a test set is slightly different. Because there are not enough perfectly constant curvature

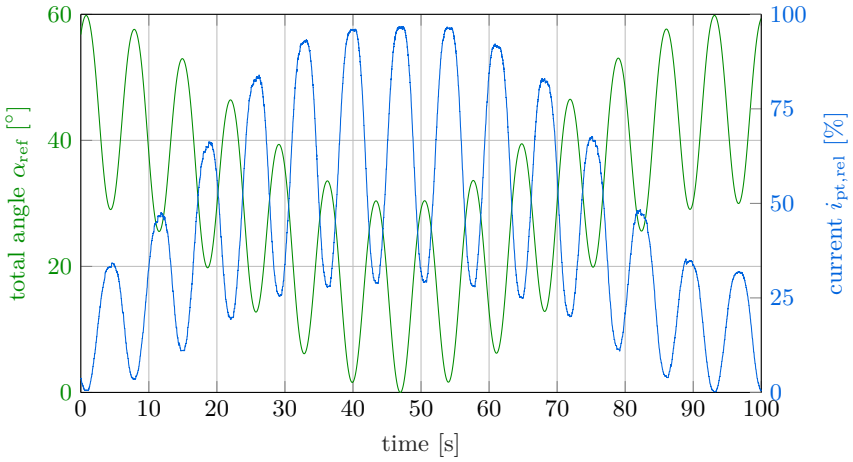


Figure 4.8: Section of measurement series for non-constant curvature.

data points in the collected data set, near constant curvature data points are also included. Here, data points for which $|\alpha_1 - \alpha_2| \leq 1^\circ$ are considered to have nearly constant curvature and are included in the training set. The remaining data points are used as the validation set. The polynomial regression is performed with a polynomial of order $k = 5$, as this polynomial order gave the best results for the case of constant curvature. The polynomial coefficients obtained are listed in Tab. 4.2.

The validation is done in the same way as for the constant curvature case. In Fig. 4.9 the relative phototransistor current is plotted over the angles α_1 and α_2. Additionally, the fifth order polynomial obtained from the calibration is plotted for the constant curvature case. It can be seen that this polynomial is a good approximation even over a wide range of non-constant curvature points. This is confirmed by the error plot in Fig. 4.10, which shows the measurement error as defined in equation (4.8) over the angles α_1 and α_2. It can be seen that for angle differences up to $\pm 10^\circ$ the error of the total angle α is mostly below 2°, which is sufficient for most soft robotic applications. Large errors of more than 10° occur only for very large angle differences of more than 20°.

This can also be seen in Fig. 4.11. Here a section of the POF measurement and the reference measurement are plotted over time. It can be clearly seen that the two measurements partly agree very well. A detailed examination shows that this is always the case if the difference between the angles α_1 and α_2 is not too large. In other parts of the measurement series the error of the POF measurement is larger. This can be explained by a larger difference between α_1 and α_2.

It can be concluded that this sensor is also suitable for determining the total angle in applications with non-constant curvature, as long as the angle difference is not too large. If the achievable accuracy is not sufficient for a particular application due to the non-constant curvature, the sensor can be divided into two or more separate sensors in series.

Table 4.2: Polynomial coefficients b_i for a polynomial order of $k = 5$ obtained by the calibration for non-constant curvature.

k	b_0	b_1	b_2	b_3	b_4	b_5
5	60.0	-0.711	$9.2 \cdot 10^{-3}$	$-1.92 \cdot 10^{-4}$	$2.14 \cdot 10^{-6}$	$-1.05 \cdot 10^{-8}$

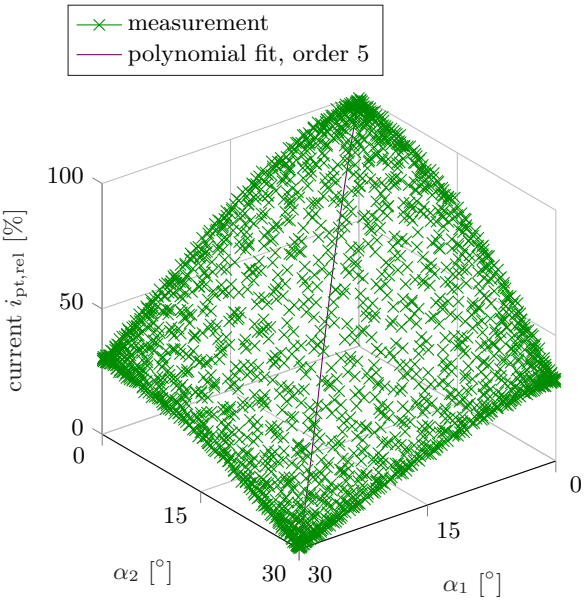


Figure 4.9: Relative phototransistor currents $i_{\mathrm{pt,rel}}$ for different angles α_1 andα_2.

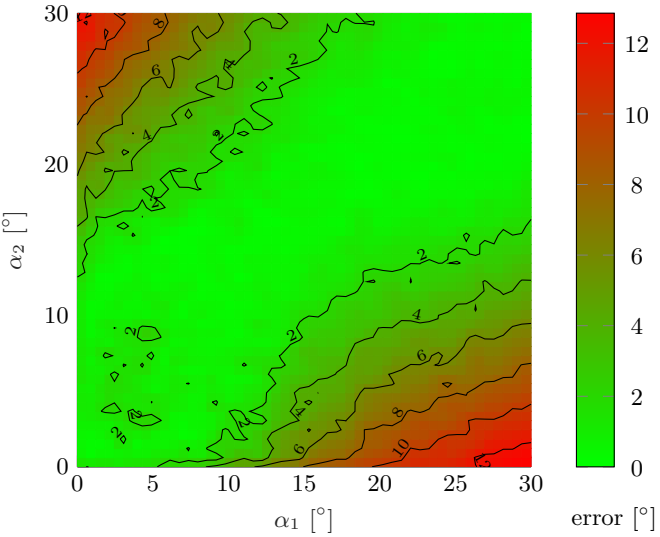


Figure 4.10: Measurement error for different angles α_1 and α_2.

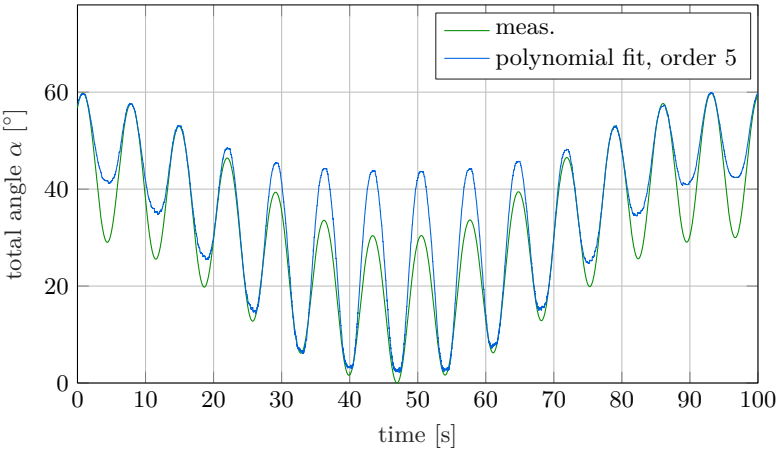


Figure 4.11: Curvature sensor measurement and reference value of the total angle for non-constant curvature.

4.1.3 Application to a Soft Gripper as a Test System

Finally, the suitability of the presented curvature sensor for soft robotic applications is investigated. As an example, the integration of three curvature sensors in the fingers of a three finger gripper for shape estimation is considered.

Design of the Gripper

The gripper is shown in Fig. 4.12. It is tendon actuated by three servos and has an opening angle of 90°. Seven AprilTags [Olson11] are attached to each of the fingers to allow camera tracking as a reference measurement. A "Logitech C270" camera is used for camera tracking. The camera tracking system determines the rotation around the bending axis for each of the AprilTags, which allows the curvature of the finger between two AprilTags to be calculated.

In Fig. 4.13 one of the fingers is shown in detail. It consists of an upper part and a lower part. The upper part is made out of foam and is needed for the tendon actuation. The lower part is made of 6 mm thick silicone of type "HT 45" and provides the required bending stiffness of the finger. The three curvature sensors are embedded one behind the other in the silicone matrix of the lower part. The positioning of the sensors is indicated in Fig. 4.13. This allows the finger to be discretized into three pieces for which the curvature can be measured. In the following, it is assumed that the curvature in each of these pieces is constant.

With this gripper, two types of grips can be achieved: enclosing grips and fingertip grips. In Fig. 4.14 enclosing and fingertip grips with one finger of the gripper

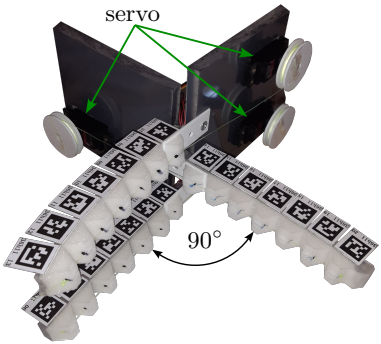


Figure 4.12: Design of the gripper.

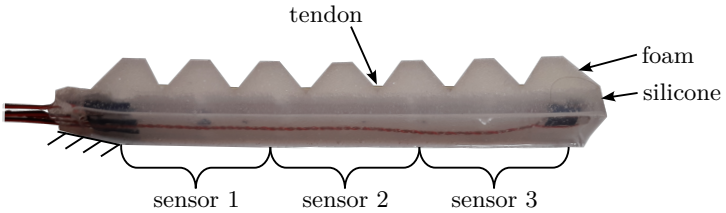


Figure 4.13: One finger of the gripper with integrated curvature sensors.

are shown for different circular objects. For enclosing grips, the finger is in contact with the object along almost its entire length. For cylindrical objects, the curvature and contact forces are nearly constant along the finger. For fingertip grips, often only the fingertip is in contact with the object. Therefore, the curvature and contact forces at the fingertip are much greater than the rest of the finger. When grasping, an enclosing grip is usually achieved first. As the tendon force increases, the grip changes to a fingertip grip.

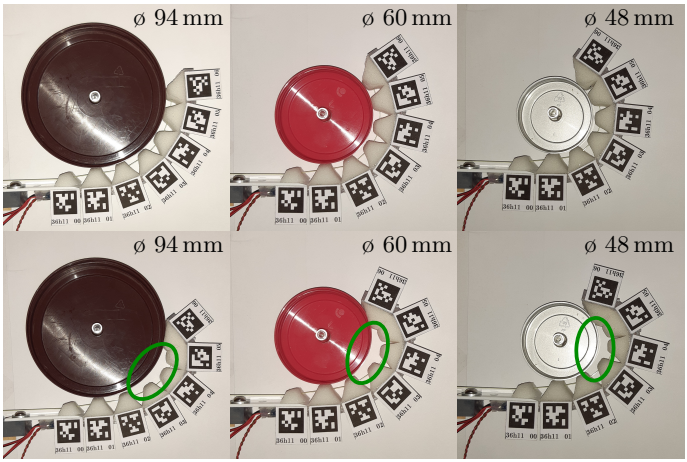


Figure 4.14: Different enclosing grips (top) and fingertip grips (bottom). The green oval marks the non-contact area in fingertip grips.

Sensor Calibration

In a first step, the sensors integrated into the fingers have to be calibrated. Calibration is performed separately for each finger. A small cylindrical object with a diameter of 48 mm is grasped with each of the fingers. The object is clamped so that it cannot be pushed away. The object size is close to the smallest object size that can be gripped by the gripper. Therefore, it can be expected that the maximum curvature occurs during gripping. Starting from the straight configuration of the finger, an enclosing grip is first performed. In the next step, the gripper is further closed until the transition to a fingertip grip is completed. The gripper is then opened again. During the grip, the relative phototransistor currents i_{pt}, as defined in Sec. 4.1.2 and the curvature along the finger obtained from the AprilTag reference measurement are recorded.

Then, for each of the sensors, a polynomial fit with a fifth order polynomial is performed as described in Sec. 4.1.2 to obtain a relationship between the relative phototransistor current i_{pt} and the bending angle α. The calibration results for only one of the fingers are shown below. The results for the other two fingers are qualitatively comparable. In Fig. 4.15 the curvature of the three curvature sensors obtained by the reference measurement is plotted against the relative phototransistor current i_{pt}. In the graph it is highlighted which part of the measurement belongs to the enclosing grip and which part belongs to the fingertip grip. The polynomial fit is also shown. It can be seen that the data from the enclosing grips fit the polynomials very well. However, the fingertip grips exhibit hysteretic behavior that cannot be represented by the

polynomial model and therefore reduces the accuracy. The hysteretic behavior can be explained by the large contact and friction forces at the fingertip. These cause deformation with non-constant curvature along the sensor. In addition, deformations of the finger other than pure bending, such as shear and strain, also occur. The cross section of the finger is also deformed. Since the markers are not placed directly in the neutral fiber of the finger where the curvature sensor is located, this causes a deviation between the curvature sensor measurement and the reference measurement. These effects are strongest at the fingertip, where the greatest external forces occur. A direct comparison of the three sensor fits shows clear differences. These can be explained by manufacturing inaccuracies. They are easily compensated for during calibration and do not affect the sensor performance.

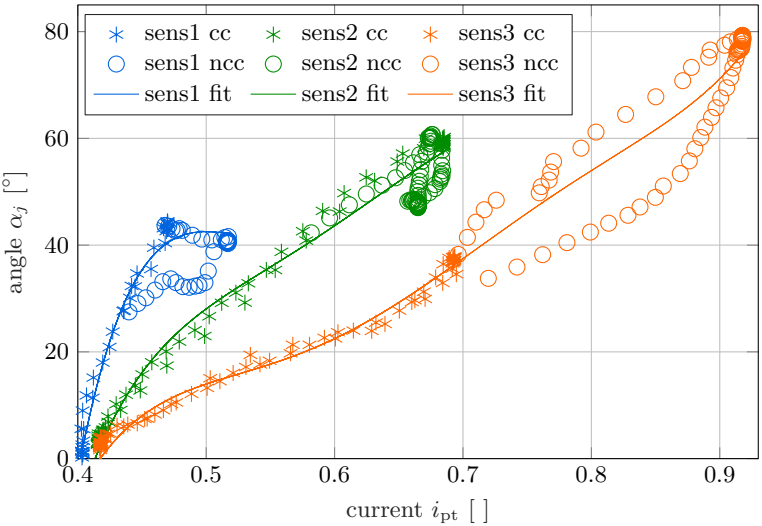


Figure 4.15: Calibration of the three curvature sensors of one finger with enclosing grip with constant curvature (cc) as well as fingertip grip with non-constant curvature (ncc) along the sensors.

Experimental Validation

To validate the calibration and the overall performance of the sensor different objects are gripped. These are three round objects with diameters of 94 mm, 60 mm and 48 mm and one oval object with a semi-minor axis of 13 mm and a semi-major axis of 20 mm. The grips of the cylindrical objects are shown in Fig. 4.14. The measurements with the POF sensor as well as the reference measurement with AprilTags are plotted over time in Fig. 4.16 for enclosing grips and in Fig. 4.17 for fingertip grips.

Altogether there is a good agreement between the POF sensor measurements and the reference measurements. Over all enclosing grips, a mean error of 2.5° with a standard deviation of 1.8° could be achieved. For the round objects a maximum error of 6° could be obtained. For the oval object, the maximum error was twice as high with 10°. This can be explained by the oval shape of the object, which leads to non-constant curvatures along the individual curvature sensors, and thus to a reduced accuracy, as explained in Sec. 4.1.2.

For the fingertip grips, the mean error is almost twice as large at 4.4° with a standard deviation of 4.7°. Again, the lower accuracy can be explained by the large forces at the fingertip, which cause non-constant curvature bending and other deformations in addition to pure bending. When the maximum possible force was applied to the tendon, the oval object was still gripped with an enclosing grip. Therefore, there is no data for a fingertip grip of the oval object. This can be explained by the oval shape and the orientation of the object during the grasp.

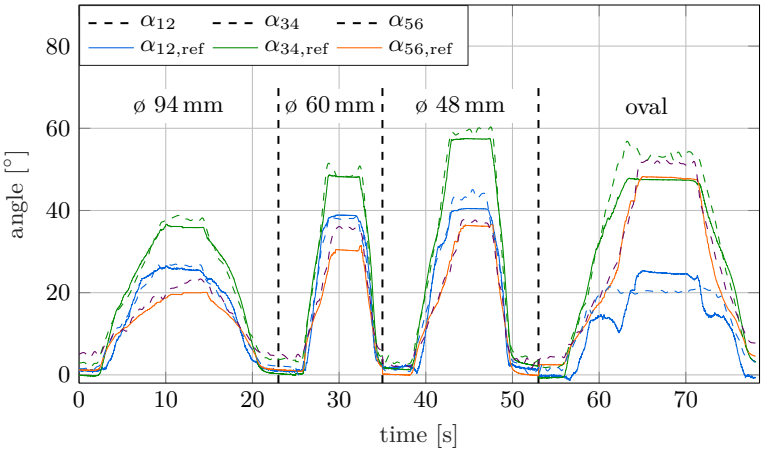


Figure 4.16: Enclosing grips with one finger of round objects with ø 94 mm, ø 60 mm, ø 48 mm and one oval object.

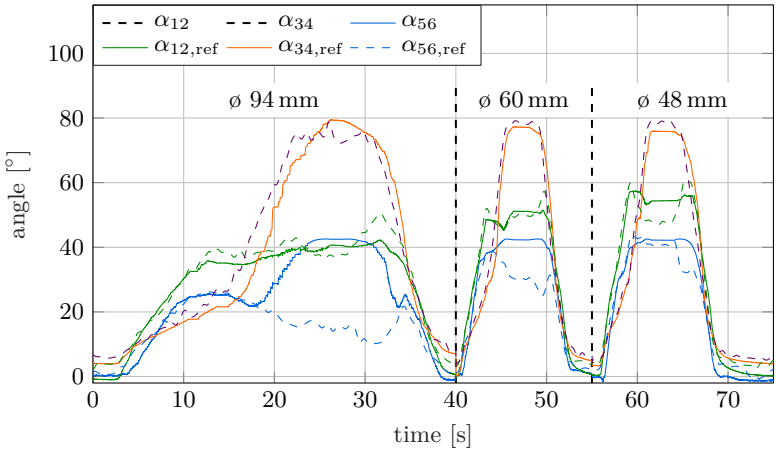


Figure 4.17: Fingertip grips with one finger of round objects with ø 94 mm, ø 60 mm, ø 48 mm.

The curvature measurements of the three individual curvature sensors integrated in each finger of the gripper can now be used to reconstruct the shape of the fingers during the grasping process. This is shown for one of the fingers in Fig. 4.18 for a slow enclosing grasp of a small cylindrical object with a diameter of 48 mm at exemplary time steps. For reference, the shape measurement of the finger using the AprilTag camera tracking system is also shown. Overall, the curvature sensor measurement and the reference measurement are in good agreement. The larger deviations between the shape measurements of the fiber optic curvature sensor and the reference measurement at the fingertip compared to the error at the center and base of the finger can be explained by the fact that the errors of the individual curvature measurements add up along the finger.

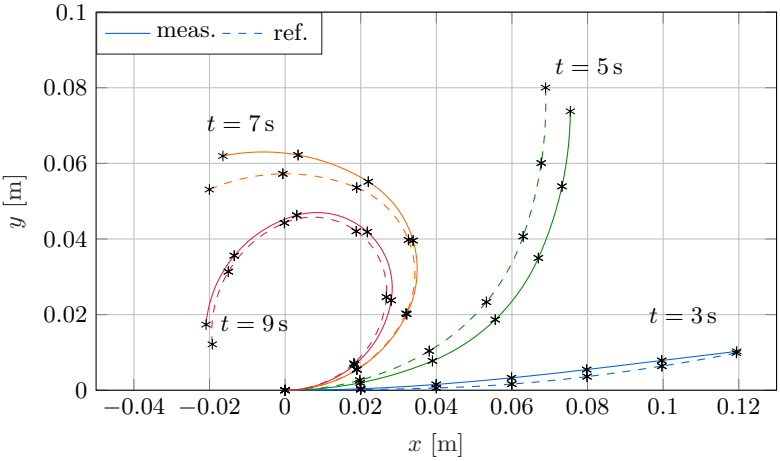


Figure 4.18: Enclosing grip with one finger of an object with ø 48 mm.

4.2 Spatial Curvature Sensor

As a second sensor, a low-cost, easy-to-fabricate fiber-optic curvature sensor for trajectory tracking control is presented that can measure curvature in two spatial directions and can be seamlessly integrated into soft robots. First, the design and fabrication of the sensor is described. Then, the application of the curvature sensor to a beam-shaped soft robot is demonstrated and the accuracy of the sensor is experimentally evaluated. The spatial curvature sensor is used in Sec. 6.1 for feedback trajectory tracking control.

4.2.1 Sensor Design

The spatial curvature sensor is an extension of the planar curvature sensor introduced in Sec. 4.1. The planar curvature sensor can measure the combined magnitude of curvature in two spatial directions, but not the direction. The spatial curvature sensor presented here can measure both the magnitude and the direction of curvature in two spatial directions. The design of the spatial curvature sensor is shown in Fig. 4.19. It consists of an infrared LED of type “SFH 4350”, four phototransistors of type “SFH 309 FA”, an incoming polymer optical fiber (POF), four segments of outgoing POF, and some connectors. The incoming POF is connected to an infrared LED which serves as a light source. The outgoing POFs are arranged symmetrically around the incoming POF so

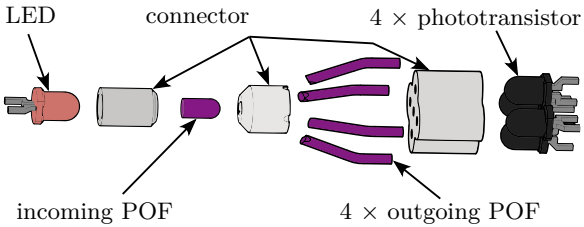


Figure 4.19: Design of the spatial curvature sensor.

that the light from the incoming POF is equally transmitted to the four outgoing POFs in a straight configuration. The angle between the incoming and outgoing POFs is 160°. This arrangement of the POFs is shown in Fig. 4.19. The tilt angle is needed to avoid dead band effects of the sensor in a straight configuration. At the other end, each of the outgoing POFs is connected to a phototransistor. In combination with an analog-to-digital converter (ADC), the phototransistors allow the measurement of the intensity Φ of the light leaving each of the four outgoing POFs. The components of the curvature sensor are held in place by three 3D-printed connectors. The incoming POF is glued to the connector in the center with transparent silicone of the type "Elastosil E43", which allows it to rotate freely. The curvature sensor can be fully embedded in the silicone matrix of soft robots. It should be placed in the neutral fiber of beam-shaped soft robots to keep the elongation of the sensor as small as possible. Although there is little infrared light in the environment, the use of colored silicone can help protect the sensor from ambient light and make it more robust in difficult lighting conditions. In Fig. 4.20, the placement of the curvature sensor in the mold of a soft robot body for silicone molding is shown.

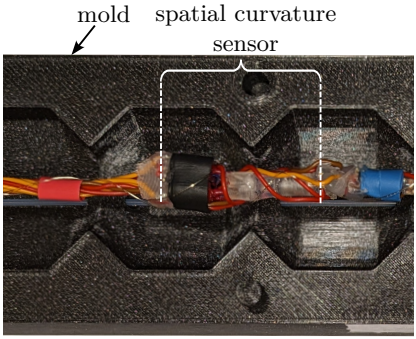


Figure 4.20: Integration of the spatial curvature sensor into the soft robot during silicone molding.

The working principle of the sensor is based on the fact that POFs transmit only light incident at an angle less than or equal to the acceptance angle θ_{acc} of the POF. If the angle of incidence is greater than θ_{acc}, there is no total reflection between the core and cladding of the POF and the light can escape from the POF. The more two successive POF segments are inclined toward each other, the greater the angle of incidence of the light. Therefore, less light is transmitted. The acceptance angle depends on the refractive indices of the core n_{core}, the cladding n_{cladding} and the surrounding medium n_0. Here, POFs with an acceptance angle of $\theta_{\mathrm{acc}} \approx 20°$ are used. The LED provides a constant light intensity. Therefore, the light intensity measured by each phototransistor depends only on the angle between the incoming POF and the outgoing POF connected to the phototransistor. The outputs of the sensor are the four phototransistor currents $\boldsymbol{i}_{\mathrm{pt}} = \begin{bmatrix} i_{\mathrm{pt},1} & i_{\mathrm{pt},2} & i_{\mathrm{pt},3} & i_{\mathrm{pt},4} \end{bmatrix}^\top$ of the phototransistors at the outgoing POFs. As with the planar curvature sensor, the measured currents of the phototransistor are normalized in the following to a value range of 0 % for the lowest measured current and 100 % for the highest measured current. If the incoming and outgoing POFs are perfectly aligned, most light can be transmitted from the incoming POF to the outgoing POF and finally reach the phototransistor. As the angle between the incoming and outgoing POF increases, less light can be transmitted and the measured intensity decreases. The measurement circuit used for intensity measurement is the same as for the planar curvature sensor described in Sec. 4.1 and shown there in Fig. 4.2.

In a final step, the curvature along the sensor can be determined from the intensity measurements. Theoretically, the angle between the incoming and outgoing POF can be determined analytically from the ratio of the light intensities measured by the four phototransistors. From this angle, the curvature can be determined. However, due to many unknown parameters and manufacturing inaccuracies, it is easier to derive this relationship directly using data-driven approaches. This is also done in the following.

4.2.2 Application to a Soft Robotic Test System

In the following, the application of the curvature sensor to a beam-shaped soft robot for tip position estimation is examined. As a test system, the soft robot introduced in Sec. 3.1 is used. Since the curvature sensor must be integrated into the soft robot during the silicone molding process, an additional version of the test system was fabricated to evaluate the sensor performance. The system inputs of the test system are the three tendon lengths $\begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^\top$, the system outputs are the x- and the y-coordinate of the tip position $\boldsymbol{z}$. The cube with AprilTags [Olson11] attached to the tip of the soft robot is used for reference measurements and calibration.

Sensor Integration and Evaluation

The spatial curvature sensor is integrated into the bottom two segments of the soft robot. The placement of the sensor during fabrication can be seen in Fig. 4.20. In the following, the relationship between the phototransistor currents $\boldsymbol{i}_{\mathrm{pt}} = \begin{bmatrix} i_{\mathrm{pt},1} & i_{\mathrm{pt},2} & i_{\mathrm{pt},3} & i_{\mathrm{pt},4} \end{bmatrix}^\top$ and the tip position $\boldsymbol{z} = \begin{bmatrix} x_{\mathrm{tip}} & y_{\mathrm{tip}} \end{bmatrix}^\top$ is learned directly with a feedforward neural network. The structure of the neural network is shown in Fig. 4.21. The inputs of the neural network are the four phototransistor currents. The outputs are the x- and y-coordinates of the tip position. The output layer uses linear activation functions. Additionally, the neural network has two hidden layers with tanh activation functions and 15 neurons each.

The training data collection is performed analogous to the training data collection for the data-driven quasi-static forward model described in Sec. 3.2.2. The robot successively approaches 350 tip position points evenly distributed over the workspace of the soft robot. For each of these points, the actual tip position is measured with the camera tracking system and the ADC measurements of the curvature sensor are recorded. Additionally, the applied tendon length $\boldsymbol{s}$ to approach this point is recorded. The collected data is split into a training set that contains 70 % of the data and a test set that contains the remaining 30 % of the data. The tip position measurements from the camera and the ADC measurements from the curvature sensor are then used to train the neural network used to evaluate the curvature sensor. A section of the collected training data is shown in Fig. 4.22. The recorded values of the tendon length $\boldsymbol{s}$ and the

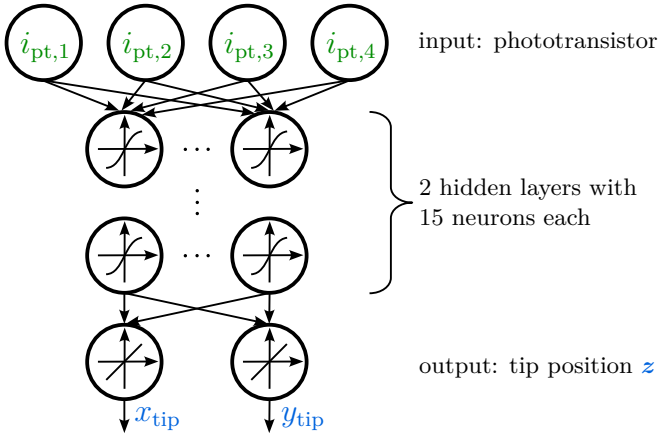
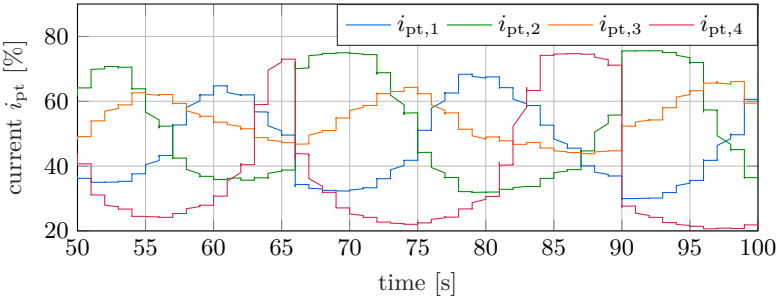
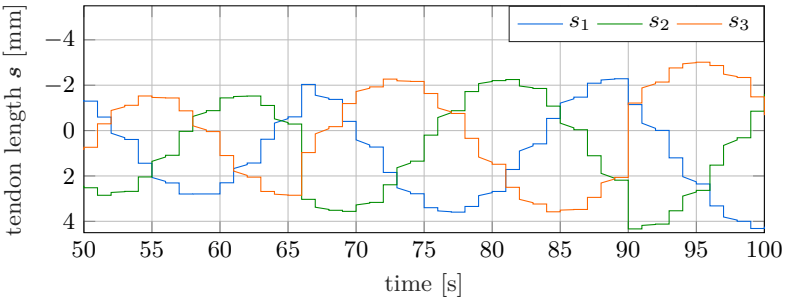


Figure 4.21: Structure of the NN to convert the raw measurement data of the four phototransistors into a tip position.

camera measurements of the tip position can be used to derive a neural network quasi-static forward model of the soft robot as described in Sec. 3.2 or a neural network quasi-static inverse model of the soft robot which is described in Sec. 5.1.



a) Phototransistor current of the curvature sensor.



b) Tendon length.

Figure 4.22: Section of the curvature sensor measurement data for the training of the neural network.

Sensor Performance

To evaluate the sensor performance, the first step is to evaluate the sensor performance on the test set. The error measure used is the root mean square (RMS) error, defined as

$$e_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{z}_{\mathrm{curv},i} - \boldsymbol{z}_{\mathrm{cam},i}||^2}, \tag{4.9}$$

where N is the number of points in the test set, $\boldsymbol{z}_{\mathrm{curv}}$ is the tip position measurement of the curvature sensor and $\boldsymbol{z}_{\mathrm{cam}}$ is the tip position obtained from the reference measurement with the camera tracking system. Here an RMS error of $e_{\mathrm{RMS}} = 4.6\,\mathrm{mm}$ with a standard deviation of $2.50\,\mathrm{mm}$ could be achieved on the test set. The error is $3.80\,\%$ of the radius of the workspace.

In a second step, the reproducibility of the sensor is investigated. A slow circular trajectory with a radius of $120\,\mathrm{mm}$ is followed and the tip position is measured with the external camera tracking system as a reference and with the integrated curvature sensor. This is repeated four times. In Fig. 4.23 the tip position measurements of the curvature sensor for all four iterations and the camera measurement are shown. For all four iterations a root mean square error of $e_{\mathrm{RMS}} = 7.0\,\mathrm{mm}$ was achieved on the circular trajectory. The plot shows that the measurements of the curvature sensor for all four iterations are very similar and hardly distinguishable. Overall, good sensor accuracy and very good repeatability were achieved.

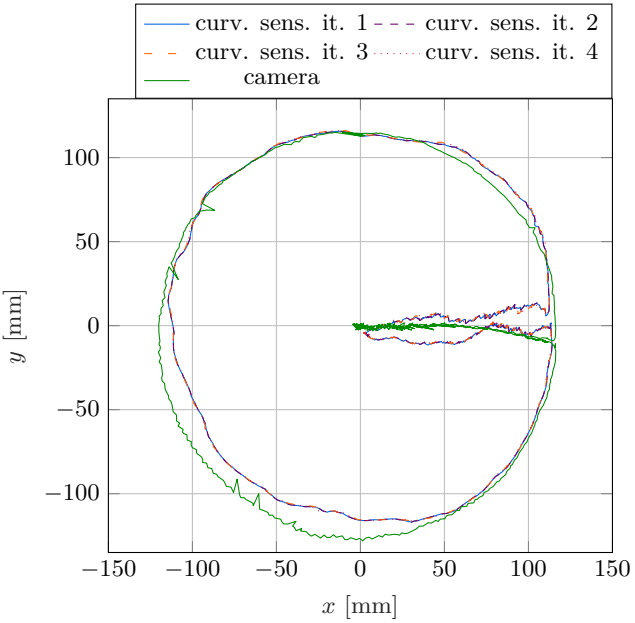


Figure 4.23: Comparison of the tip position measurement of the curvature sensor and the camera on a circular trajectory for four iterations.

Feedforward Control

With the emergence of new and more advanced soft robotic applications, control requirements increase. In particular, agility and accuracy become more important. Due to the softness and the typically large occurring strains, sensor integration into soft robots is often difficult. In practical applications, external sensors such as camera tracking systems, which are popular in research, are often not an option because they limit the flexibility of the soft robotic system. This makes feedforward control particularly important for soft robots. Feedforward controllers can be used alone or in combination with a feedback controller. This reduces the requirements for the feedback controller and the amount of sensor data needed. Because soft robots can usually deform continuously, they often have a very large number of degrees of freedom and are therefore underactuated. They are also often redundantly actuated. In addition, modeling is usually difficult due to large elastic deformations, unknown material parameters, and manufacturing inaccuracies. This makes soft robots much more challenging to control than rigid robots.

In soft robotics, a distinction is made between quasi-static and dynamic controllers. Controllers are considered quasi-static when inertia and damping properties of the soft robot are neglected. In the soft robotics literature, the terms “quasi-static” and “kinematic” are often used interchangeably. A further distinction is made between model-based and model-free controllers. Note that in soft robotics, controllers based on data-based models are usually considered model-free [ThuruthelEtAl18a].

Quasi-static controllers are the most commonly used controllers in soft robotics [ThuruthelEtAl18a]. However, because they neglect the dynamics, they are limited to slow motions to avoid oscillations. Most model-based quasi-static controllers are based on direct inversion of the kinematics of the model, see e.g. [CamarilloEtAl09]. When this is not uniquely possible, differential inverse kinematics [BaillyAmirat05, JonesWalker06, RendaEtAl22] is often used. For highly nonlinear systems, soft robots with large manufacturing inaccuracies, and soft robots that are difficult to model, model-free quasi-static controllers are widely used. Here, the mapping between system output and system input is usually learned directly by a neural network, see e.g. [GiorelliEtAl13, ThuruthelEtAl16b, ThuruthelEtAl16a].

When more agile motion is required, quasi-static controllers are not sufficient and dynamic controllers must be used. Often, predictive control approaches are used. These are based on different dynamic models such as nonlinear autoregressive networks with exogenous inputs (NARX) models [ThuruthelEtAl17b, ThuruthelEtAl18b], models derived from Koopman operator-based system identification [BruderEtAl21], and models based on spectral submanifold reduction (SSMR) [AloraEtAl23]. Reinforcement learning, mostly based on deep Q learning, is also applied, see e.g. [ZhangEtAl17, SatheeshbabuEtAl19].

In this chapter, different feedforward control approaches for soft robots are presented and compared experimentally. First, in Sec. 5.1 a neural network based quasi-static control approach is presented. Second, in Sec. 5.2 a model-based dynamic control approach based on servo-constraints is shown. Third, in Sec. 5.3 a hybrid control approach, that is a combination of both approaches, is presented. This allows for an accurate data-driven representation of the inverse quasi-static behavior of the system while preserving a data-efficient model-based inverse dynamic model. Finally, in Sec. 5.4 experimental results for the trajectory tracking control performance of the three control approaches are presented. For this purpose, the three controllers are applied to the test system introduced in Sec. 3.1. Large parts of this chapter have also been published in [GrubeEtAl24].

5.1 Neural Network based Quasi-Static Inverse Model

For feedforward quasi-static control of the soft robot, an inverse version of the quasi-static NN-based forward model presented in Sec. 3.2 can be used. Since this model does not contain any redundancy, direct unique inversion is possible. There are several different approaches to determine the inverse of the quasi-static model, one of which is to invert the quasi-static model by solving the nonlinear equation $\boldsymbol{z}_{\mathrm{des}} = \boldsymbol{z}(\boldsymbol{s})$ numerically, e.g. using the MATLAB function `fmincon`. Here $\boldsymbol{z}_{\mathrm{des}}$ is the desired tip position and the relationship $\boldsymbol{z}(\boldsymbol{s})$ between tip position $\boldsymbol{z}$ and tendon length $\boldsymbol{s}$ is represented by the NN forward model.

However, a more efficient approach is to directly learn an inverse version of the NN by using the system input as the output of the NN and the system output as the input of the NN, as suggested, for example, in [GiorelliEtAl13]. The structure of the resulting controller is shown in Fig. 5.1. This results in a feedforward neural network with the tip position $\boldsymbol{z} = \begin{bmatrix} z_x & z_y \end{bmatrix}^\top$ as an input,

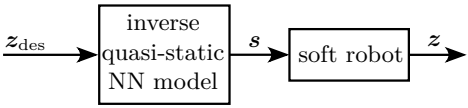


Figure 5.1: Structure of the quasi-static NN controller.

the tendon length $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^\top$ as an output and n_{layer} hidden layers with n_{neuron} neurons each. The hidden layers have a tanh activation function, the output layer has a linear activation function. The structure of the NN is shown in Fig. 5.2. As for the quasi-static forward model described in Sec. 3.2, the redundancy of the actuation is resolved by constraining the system input $\boldsymbol{s}$ so that the sum of the tendon forces $\hat{F} = \sum F_k = \text{const}$ is constant. This is described in detail in Sec. 3.1.2. To train the neural network, the training data collected for the forward NN model is used, the training data collection process is described in Sec. 3.2.2. In the following, the choice of hyperparameters is investigated first. Then, the performance of the neural network based quasi-static inverse model is investigated in terms of accuracy and computational efficiency. The procedure is analogous to that described in Secs. 3.2.3 and 3.2.4 for the forward model and is therefore only briefly described here.

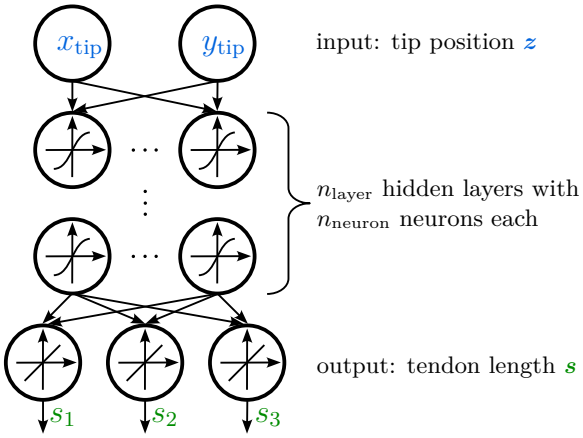


Figure 5.2: Structure of the neural network used to represent the inverse quasi-static behavior.

5.1.1 Hyperparameters of the Neural Network

As for the forward model described in Sec. 3.2.3, the most important hyperparameters of the neural network are the required number of hidden layers n_{layer} and the number of neurons n_{neuron} in each of these layers, both of which have to be determined heuristically. Additionally, the number of training samples $n_{\text{samples,train}}$ needed to train the neural network is an important parameter that determines the overall training effort. These three parameters are investigated in the following.

In a first step, different neural networks with $n_{\text{layer}} = 1 \dots 5$ hidden layers and $n_{\text{neuron}} = 1 \dots 50$ neurons per layer are trained and compared with respect to the root mean square error e_{RMS} of the neural network output, i.e. the tendon length, of the validation set. It is defined as

$$e_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i}^{N} ||\boldsymbol{s}_{\text{sim}} - \boldsymbol{s}_{\text{exp}}||^2}, \tag{5.1}$$

where N is the number of samples, $\boldsymbol{s}_{\text{exp}}$ is the tendon length taken from the experiment, and $\boldsymbol{s}_{\text{sim}}$ is the tendon length obtained from the inverse NN model. For reference, the tendon length error is compared to the maximum length change $\Delta\ell_{\text{max}} = \max(s_k) - \min(s_k) = 30\,\text{mm}$ of each of the tendons in the workspace. The results are shown in Fig. 5.3. As for the forward NN model presented in Sec. 3.2, the best performance can be obtained for $n_{\text{neuron}} \approx 10$ neurons per layer and $n_{\text{layer}} = 2$ hidden layers. For less than $n_{\text{neuron}} = 4$ neurons per layer, the error e_{RMS} increases strongly, which can be explained by underfitting. For more than $n_{\text{neuron}} = 16$ neurons per layer, the error e_{RMS} remains constant or even increases slightly, which can be explained by overfitting. The number of hidden layers n_{layer} has hardly any influence. The only notable difference is that for a medium number of $4 \dots 15$ neurons per layer, the networks with one hidden layer perform slightly worse than the other networks. Therefore, in the following, a network with $n_{\text{layer}} = 2$ hidden layers and $n_{\text{neuron}} = 10$ neurons per layer is used, as for the forward model presented in Sec. 3.2.

Finally, the required size of the training set is examined. Therefore, the neural network with $n_{\text{layer}} = 2$ hidden layers and $n_{\text{neuron}} = 10$ neurons per layer is trained with different numbers of samples of $n_{\text{sample}} = 1 \dots 646$ in the training set. The results are shown in Fig. 5.4. It can be seen that $n_{\text{samples}} \approx 55$ samples in the training set are required to achieve a tendon length error of less than 1 % of the maximum tendon length change $\Delta\ell_{\text{max}}$. For more than $n_{\text{samples}} \approx 150 \dots 200$ samples, the error decreases only slightly further. In the following, as for the forward model, a training set of 150 samples is used. This corresponds to a tendon length error of 0.3 % of the maximum tendon length change $\Delta\ell_{\text{max}}$. Together with the validation and test sets, a total of ≈ 215 data points are required, assuming a split of 70 % of the data in the training set and 15 % each of the data in the

validation and test sets. On the physical robot, this data can be collected in about 5 min.

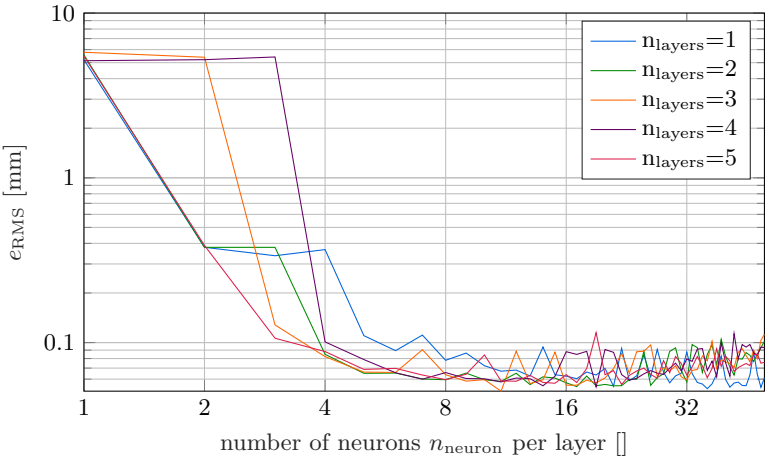


Figure 5.3: Influence of the number of layers n_{layer} and the number of neurons per layer n_{neurons} on the accuracy of the NN quasi-static inverse model.

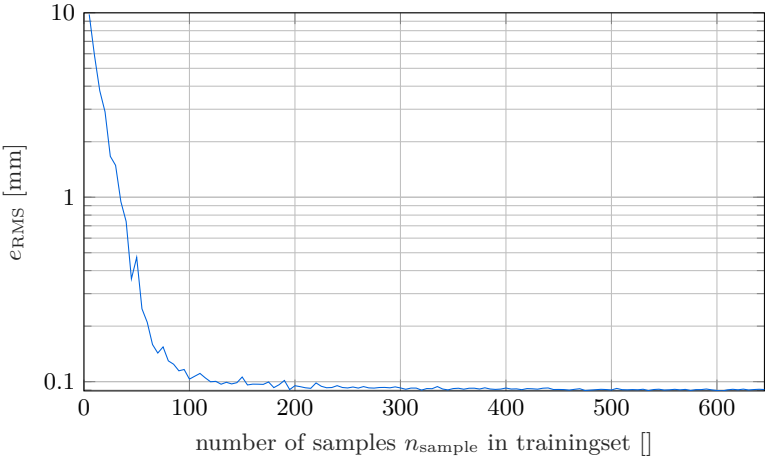


Figure 5.4: Influence of the training set size $n_{\mathrm{samples,train}}$ on the accuracy of the NN quasi-static inverse model.

5.1.2 Model Performance

To evaluate the performance of the quasi-static NN model, a neural network with $n_{\text{layer}} = 2$ hidden layers with $N_{\text{neuron}} = 10$ neurons each is trained using a training set of $n_{\text{sample}} = 150$ samples and a validation set of 32 samples. As a test set, the full test set collected in Sec. 3.2.3 with 139 samples is used.

In Fig. 5.5, the tendon lengths $s_1 \dots s_3$ taken from the experiment and obtained from the inverse model are plotted in a 3D plot. A root mean square error of $e_{\text{RMS}} = 0.11\,\text{mm}$ and a standard deviation of 0.13 mm was obtained on the test set. The error is less than 0.36 % of the maximum tendon length change $\Delta\ell_{\text{max}}$. As for the forward model, the error is mainly due to a slight hysteretic behavior of the soft robot, resulting from cable friction in the tendon guides, which cannot be represented in this type of model. There are a few outliers with an error of up to 1.4 mm, which can be attributed to disturbances during the measurement recording. Overall, a very good accuracy was achieved, which is probably sufficient for most soft robotic applications. The main limitation of this model is that it does not include dynamics, which is only an issue when fast motions are required.

Finally, the computational efficiency of the neural network is evaluated. For this, the test set is evaluated by the neural network sample by sample 10 000 times, resulting in an average evaluation time of $T_{\text{comp}} = 2.3\,\mu\text{s}$ per sample for the compiled version of the neural network. Without compilation, the average evaluation time increases to a value of $T_{\text{uncomp}} = 14.9\,\mu\text{s}$. In both cases, the computation time is several orders of magnitude shorter than typically required for practical applications.

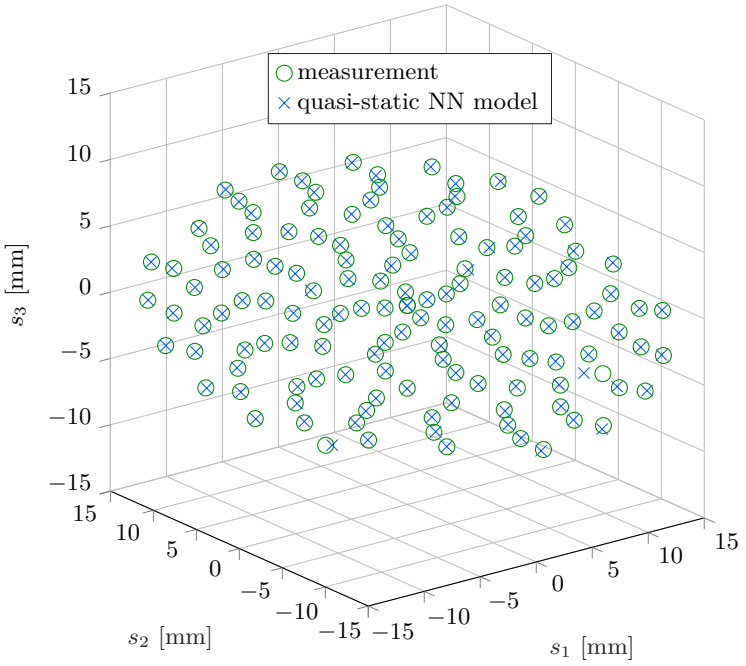


Figure 5.5: Tendon length $s_1 \dots s_3$ from experiment and simulation using the quasi-static NN model on the test set.

5.2 Model-Based Dynamic Inverse Model using Servo-Constraints

The servo-constraints approach is a model inversion approach known from multi-body dynamics. In the following, this approach is used to derive an inverse dynamic model for the soft robotic test system based on a dynamic PCC forward model. First, the derivation of the PCC forward model for the soft robotic test system presented in Sec. 3.1 is described. Second, the servo-constraints framework used to invert the dynamic forward model is presented. Finally, the application of the servo-constraints framework to the dynamic forward model of the test system is shown.

5.2.1 Forward Model

To derive an inverse dynamic model using the servo-constraints approach, a dynamic forward model is required in a first step. To account for the typically large amounts of data required for data-driven dynamic models, a model-based approach is chosen. A PCC model is chosen because it is the simplest of the modeling approaches presented in chapter 2. In order to allow a simpler handling of singularities compared to the parametrization of the PCC model introduced in Sec. 2.1, a parametrization based on [AllenEtAl20] is chosen. Both parametrizations differ in the choice of generalized coordinates. In the parametrization presented in Sec. 2.1, the projection of the bending in x- and y-direction is chosen as generalized coordinates. In the parametrization presented here, based on [AllenEtAl20], the x-and y-components of the rotation axis of each piece are chosen as generalized coordinates. Since the test system is stiff in longitudinal direction and no torsion occurs, only bending in two spatial directions is included in the model. In the following, the discretization and parametrization of the PCC model is described below. Then the application of the parametrization to the soft robotic test system introduced in Sec. 3.1 is shown. Next, the derivation of the forces and moments is described. Finally, the derivation of the equations of motion is shown.

Discretization and Parametrization of the PCC Model

To describe the deformation, the soft robot is discretized into N_{pieces} pieces. The parametrization of a PCC piece is shown in Fig. 5.6. The coordinate system K_i is attached to the top of piece i, the coordinate system K_{i-1} is attached to the bottom of piece i. The coordinate system K_0 at the bottom of the first piece is identical to the global coordinate system K. The coordinate u describes the relative position of a section along the centerline of the piece. Here, $u = 0$ describes the start of the piece and $u = 1$ describes the tip of the piece. The deformation of the piece is described as a rotation around the axis

$$\boldsymbol{w}_i = \begin{bmatrix} \mu_i & \nu_i & 0 \end{bmatrix}^\top \tag{5.2}$$

which runs through the point P_i with the position vector

$$\boldsymbol{\rho}_i = \frac{\ell_i}{\theta_i^2} \begin{bmatrix} \nu_i & -\mu_i & 0 \end{bmatrix}^\top \tag{5.3}$$

with the total bending angle

$$\theta_i = ||\boldsymbol{w}_i|| = \sqrt{\mu_i^2 + \nu_i^2}. \tag{5.4}$$

The x_i- and y_i- coordinates μ_i and ν_i of the rotation axis $\boldsymbol{w}_i$ are chosen as the generalized coordinates.

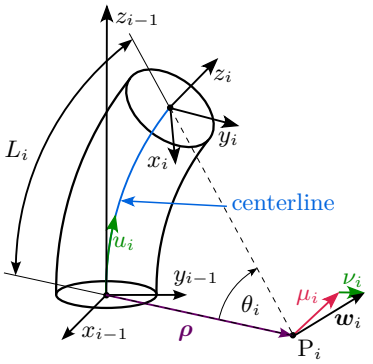


Figure 5.6: Parametrization of a PCC piece.

With this parametrization the position vector $\boldsymbol{r}_{i,u,\mathrm{loc}}$ of a cross section on the centerline of piece i at position u in local coordinates is

$$\boldsymbol{r}_{i,u} = \begin{bmatrix} \rho_{i,x}\sigma_{i,u} \\ \rho_{i,y}\sigma_{i,u} \\ ||\boldsymbol{\rho}_i|| \sin(\theta_{i,u}) \end{bmatrix} \tag{5.5}$$

with $\theta_{i,u} = u\theta_i$ and $\sigma_{i,u} = \cos(\theta_{i,u}) - 1$. The rotation matrix $\boldsymbol{R}_{\mathrm{K}_{i-1}\mathrm{K}_{i,u}}$, which describes the rotation from a cross section of piece i at position u to the coordinate system K_{i-1} at the bottom of piece i results in

$$\boldsymbol{R}_{\mathrm{K}_{i-1}\mathrm{K}_{i,u}} = \begin{bmatrix} \sigma_{i,u}\tilde{\nu}_i^2 + 1 & -\sigma_{i,u}\tilde{\mu}_i\tilde{\nu}_i & \tilde{\nu}_i \sin(\theta_{i,u}) \\ -\sigma_{i,u}\tilde{\mu}_i\tilde{\nu}_i & \sigma_{i,u}\tilde{\mu}_i^2 + 1 & -\tilde{\mu}_i \sin(\theta_{i,u}) \\ -\tilde{\nu}_i \sin(\theta_{i,u}) & \tilde{\mu}_i \sin(\theta_{u,i}) & \cos(\theta_{u,i}) \end{bmatrix} \tag{5.6}$$

with $\tilde{\nu}_i = \frac{\nu_i}{\theta_i}$ and $\tilde{\mu}_i = \frac{\mu_i}{\theta_i}$.

The position vector $\boldsymbol{r}_{i,u}$ in global coordinates of the cross section of piece i at position u and the rotation matrix $\boldsymbol{R}_{\mathrm{KK}_{i,u}}$ from this cross section to the global coordinate system result in

$$\boldsymbol{R}_{\mathrm{KK}_{i,u}} = \left(\prod_{k=1}^{i-1} \boldsymbol{R}_{\mathrm{K}_{k-1}\mathrm{K}_{k,1}} \right) \boldsymbol{R}_{\mathrm{K}_{i-1}\mathrm{K}_{i,u}}, \tag{5.7}$$

$$\boldsymbol{r}_{i,u} = \left(\sum_{k=1}^{i-1} \boldsymbol{R}_{\mathrm{KK}_{k,1}} \boldsymbol{r}_{k,1,\mathrm{loc}} \right) + \boldsymbol{R}_{\mathrm{KK}_{i,u}} \boldsymbol{r}_{i,u,\mathrm{loc}}. \tag{5.8}$$

Application of the Discretization to the Soft Robotic Test System

The soft robot introduced in Sec. 3.1 and used here as a test system is discretized into $N_{\mathrm{pieces}} = 1$ pieces. This results in 2 degrees of freedom and is sufficient

to capture the dominant dynamics of the system. However, an extension to finer discretizations is straightforward. To accurately derive the internal forces and the inertia of the soft robotic test system, the soft robot piece is divided into $N_{\text{subpieces}} = 3$ subpieces of equal length of $\ell_i = 60\,\text{mm}$, as shown in Fig. 5.7. In the following, the subpieces are denoted by the index j. The mass and inertia properties of each subpiece are concentrated in the center of gravity of each subpiece, which is located at u_j. The cube on top of the soft robot is added to the top segment. The stiffness and damping properties are distributed among the elastic links between them. The center of gravity positions $u_{\text{cg},j}L$, masses m_j and mass moments of inertia $\hat{I}_{xx,j}, \hat{I}_{yy,j}, \hat{I}_{zz,j}$ of the three subpieces are derived from a CAD model of the soft robot and are listed in Tab. 5.1. Here $L = 180\,\text{mm}$ is the total length of the soft robot. The bending stiffness $EI_{xx} = 18.3 \cdot 10^{-3}\,\text{Nm}^2$ and the specific damping constant $\frac{d}{L} = 56 \cdot 10^{-3}\,\text{Nm}^2\text{s}$ are derived experimentally and assumed to be constant along the beam. Note that I_{xx} is the second moment of area, and $\hat{I}_{xx}$ is the mass moment of inertia.

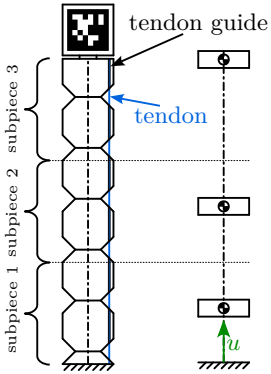


Figure 5.7: Discretization of the soft robot into one piece and 3 subpieces.

Table 5.1: Center of gravity position $u_{\mathrm{cg}}L$, mass m and mass moments of inertia $\hat{I}_{xx}$, $\hat{I}_{yy}$, $\hat{I}_{zz}$ of the three subpieces of the soft robot.

	$u_{\mathrm{cg}}L$ [mm]	m [kg]	$\hat{I}_{xx}$ [kgm^2]	$\hat{I}_{yy}$ [kgm^2]	$\hat{I}_{zz}$ [kgm^2]
subpiece 1	32	0.04	13.96	13.96	3.99
subpiece 2	92	0.04	13.96	13.96	3.99
subpiece 3	178.6	0.095	70.04	70.04	9.88

Internal and External Forces and Moments

The internal forces and moments are derived following Sec. 2.1.2. For simplicity, linear material behavior is assumed. As shown in [GrubeSeifried22b], this is a sufficiently good approximation for most deformations of soft robots. The internal moments $\boldsymbol{\ell}_{i,j,\mathrm{bnd}}$ resulting from the bending stiffness of the individual subpieces acting on the individual subpieces result in

$$\boldsymbol{\ell}_{i,j,\mathrm{bnd}} = \ell_{i,j,\mathrm{bnd}} \boldsymbol{R}_{\mathrm{KK}_{i,u_j}} \begin{bmatrix} -\sin(\varphi_i) & \cos(\varphi_i) & 0 \end{bmatrix}^\top \tag{5.9}$$

with the magnitude of the internal moments

$$\ell_{i,j,\mathrm{bnd}} = EI_{xx,\mathrm{loc}} \frac{\sqrt{\mu_i^2 + \nu_i^2}}{\ell_i} \tag{5.10}$$

and the angle of the bending plane

$$\varphi_i = \operatorname{atan2}(\nu_i, \mu_i). \tag{5.11}$$

The internal damping moments $\boldsymbol{\ell}_{i,j,\mathrm{dmp}}$ acting on the individual subpieces result in

$$\boldsymbol{\ell}_{i,j,\mathrm{dmp}} = \begin{cases} d(\boldsymbol{\omega}_{\mathrm{KK}_{i,j}}) & j = 1 \\ d(\boldsymbol{\omega}_{\mathrm{KK}_{i,j}} - \boldsymbol{\omega}_{\mathrm{KK}_{i,j-1}}) & \text{otherwise.} \end{cases} \tag{5.12}$$

Here, the angular velocity $\boldsymbol{\omega}_{\mathrm{KK}_{i,j}}$ in the global coordinate frame can be calculated as

$$\boldsymbol{\omega}_{\mathrm{KK}_{i,j}} = \left(\sum_{k=1}^{i-1} \boldsymbol{R}_{\mathrm{KK}_{k,1}} \boldsymbol{\omega}_{\mathrm{K}_{k-1}\mathrm{K}_{k,1}} \right) + \boldsymbol{R}_{\mathrm{KK}_{i,u_j}} \boldsymbol{\omega}_{\mathrm{K}_{i-1}\mathrm{K}_{i,u_j}} \tag{5.13}$$

with the angular velocity $\boldsymbol{\omega}_{\mathrm{K}_{i-1}\mathrm{K}_{i,u}}$ in the local coordinate system $\mathrm{K_i}$

$$\boldsymbol{\omega}_{\mathrm{K}_{i-1}\mathrm{K}_{i,u}} = u \frac{2\mu_i\dot{\mu}_i + 2\nu_i\dot{\nu}_i}{2\sqrt{\mu_i^2 + \nu_i^2}} \begin{bmatrix} \cos(\varphi_i) \\ -\sin(\varphi_i) \\ 0 \end{bmatrix}. \tag{5.14}$$

Finally, the forces $\boldsymbol{f}_{i,j,\mathrm{grav}}$ resulting from gravity acting in negative z-direction are added resulting in

$$\boldsymbol{f}_{i,j,\mathrm{grav}} = \begin{bmatrix} 0 & 0 & -m_j g \end{bmatrix}^\top . \tag{5.15}$$

The forces and moments resulting from tendon actuation are derived as described in Sec. 2.6 and can be written in the form

$$\boldsymbol{f}_{i,j,\mathrm{act}} = \boldsymbol{B}_{i,j,\mathrm{f}}(\boldsymbol{y})\boldsymbol{F} \tag{5.16}$$

$$\boldsymbol{\ell}_{i,j,\mathrm{act}} = \boldsymbol{B}_{i,j,\ell}(\boldsymbol{y})\boldsymbol{F} \tag{5.17}$$

where $\boldsymbol{y}$ are the generalized position coordinates of the system, $\boldsymbol{B}_{i,j,\mathrm{f}}$ and $\boldsymbol{B}_{i,j,\ell}$ are the input distribution matrices derived in Sec. 2.6 and $\boldsymbol{F}$ is the vector of tendon forces.

Equations of Motion of the Forward Model

For the derivation of the equations of motion the generalized coordinates $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\mu}^\top & \boldsymbol{\nu}^\top \end{bmatrix}^\top \in \mathbb{R}^{2N_{\mathrm{pieces}}}$ are chosen. The equations of motion can now be obtained by following the Newton-Euler formalism. They can be written as a second order ODE in control affine form as

$$\boldsymbol{M}(\boldsymbol{y},t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) = \boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{F}(t). \tag{5.18}$$

Here, $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{t}) \in \mathbb{R}^{2\times 2}$ is the mass matrix, $\boldsymbol{k}(\boldsymbol{y}) \in \mathbb{R}^2$ summarizes the Coriolis, centrifugal and gyroscopic forces and $\boldsymbol{q}(\boldsymbol{y}) \in \mathbb{R}^2$ represents the internal forces and torques. The tendon forces used for actuation are summarized in the vector $\boldsymbol{F}(t) = \begin{bmatrix} F_1(t) & F_2(t) & F_3(t) \end{bmatrix}^\top \in \mathbb{R}^3$, the input distribution matrix $\boldsymbol{B}(\boldsymbol{y}) \in \mathbb{R}^{2\times 3}$ provides a mapping between the tendon forces $\boldsymbol{F}$ and the generalized coordinates. As discussed in Sec. 2.6, this system is redundantly actuated.

In order to be able to apply the servo-constraints approach for model inversion, the redundancy of the actuation of the system must be removed. This can be done either by explicitly applying additional constraints to the system or by reducing the number of system inputs. In the following, the redundancy of the system is resolved using the projection approach described in Sec. 3.1.2. To recapitulate, in a first step, a reduced system input $\boldsymbol{F}_{\mathrm{red}} = \begin{bmatrix} F_{\mathrm{red},x} & F_{\mathrm{red},y} \end{bmatrix}^\top$ with

$$\underbrace{\begin{bmatrix} F_{\mathrm{red},x} \\ F_{\mathrm{red},y} \end{bmatrix}}_{\boldsymbol{F}_{\mathrm{red}}} = \underbrace{\begin{bmatrix} 1 & -1 & -1 \\ 0 & 1 & -1 \end{bmatrix}}_{\boldsymbol{\Phi}} \underbrace{\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}}_{\boldsymbol{F}} \tag{5.19}$$

is defined. Here, $\boldsymbol{\Phi}$ is a projection matrix and the reduced system input $\boldsymbol{F}_{\mathrm{red}}$ can be interpreted as a scaled projection of the three tendon forces $\boldsymbol{F}$ in x- and

y-direction of the global coordinate frame. Note that for simplicity, the projection matrix $\boldsymbol{\Phi}$ is a scaled version of the projection matrix introduced in equation (3.1). This results in a scaled version of $\boldsymbol{F}_{\mathrm{red}}$, but otherwise has no effect on the system behavior or performance. The equations of motion of the system with the reduced system input $\boldsymbol{F}_{\mathrm{red}}$ can then be written as

$$\boldsymbol{M}(\boldsymbol{y},t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) = \boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{\Phi}^{\dagger}\boldsymbol{F}_{\mathrm{red}}(t) \tag{5.20}$$

with the pseudoinverse $\boldsymbol{\Phi}^{\dagger}$ of the projection matrix $\boldsymbol{\Phi}$

$$\boldsymbol{\Phi}^{\dagger} = \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{1}}^{\dagger} & \boldsymbol{\Phi}_{\boldsymbol{2}}^{\dagger} \end{bmatrix}, \tag{5.21}$$

$$\boldsymbol{\Phi}_{\boldsymbol{1}}^{\dagger} = \begin{cases} \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 0 \end{bmatrix} & F_{\mathrm{red},x} \leq 0 \\ \text{otherwise} \end{cases}, \tag{5.22}$$

$$\boldsymbol{\Phi}_{\boldsymbol{2}}^{\dagger} = \begin{cases} \begin{bmatrix} -1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} & F_{\mathrm{red},y} \leq 0 \\ \text{otherwise} \end{cases}. \tag{5.23}$$

5.2.2 Servo-constraints Framework for Model Inversion

The servo-constraints framework is a systematic approach for model inversion introduced to multibody dynamics in [BlajerKołodziejczyk04]. The approach is motivated by the description of multibody systems in redundant coordinates. Similar to the geometric constraints used there, in the servo-constraints approach, the equations of motion are extended by algebraic constraints

$$\boldsymbol{s}(\boldsymbol{y},t) = \boldsymbol{z}(\boldsymbol{y}) - \boldsymbol{z}_{\mathrm{des}}(t) = 0, \tag{5.24}$$

which constrain the system output $\boldsymbol{z}$ to be identical to the desired output trajectory $\boldsymbol{z}_{\mathrm{des}}$. This results in the DAE

$$\boldsymbol{v} = \dot{\boldsymbol{y}} \tag{5.25}$$

$$\boldsymbol{M}(\boldsymbol{y},t)\dot{\boldsymbol{v}} = \boldsymbol{q}(\boldsymbol{y},\boldsymbol{v},t) - \boldsymbol{k}(\boldsymbol{y},\boldsymbol{v},t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{u} \tag{5.26}$$

$$\boldsymbol{s}(\boldsymbol{y},t) = \boldsymbol{0} \tag{5.27}$$

which usually have to be solved numerically. The unknowns are the generalized position coordinates $\boldsymbol{y}$, the generalized velocity coordinates $\boldsymbol{v}$ and the system input $\boldsymbol{u}$. The desired system output $\boldsymbol{z}_{\mathrm{des}}$ becomes the input of the inverse system (5.25) to (5.27). Note that the system inputs $\boldsymbol{u}$ are not necessarily orthogonal to the constraint manifold, unlike the Lagrange multipliers $\boldsymbol{\lambda}$, which enforce the geometric constraints when describing multibody systems with redundant coordinates.

For most mechanical systems, the resulting servo constraint DAE (5.25) to (5.27) has a DAE index of at least 3 [Campbell95]. Standard solvers such as the

MATLAB solvers `ode15s` and `ode23s` typically can only solve index 1 DAEs. Therefore, index reduction techniques have to be applied. One popular choice for index reduction is minimal extension. Here, index reduction is achieved by differentiating the algebraic equation $\boldsymbol{s}(\boldsymbol{y}, t)$ until a DAE index of 1 is achieved [KunkelMehrmann04].

5.2.3 Inverse Dynamic Controller for the test System

The servo-constraints concept introduced above can now be used to set up an inverse dynamic controller for the test system. To derive the inverse model of the test system, in a first step for index reduction the constraint equation (5.24) is differentiated once resulting in

$$\dot{\boldsymbol{s}}(\boldsymbol{y}, \boldsymbol{v}, t) = \dot{\boldsymbol{z}}(\boldsymbol{y}, \boldsymbol{v}) - \dot{\boldsymbol{z}}_{\mathrm{des}}(t) = \boldsymbol{0}. \tag{5.28}$$

This can also be written as

$$\dot{\boldsymbol{s}}(\boldsymbol{y}, \boldsymbol{v}, t) = \underbrace{\frac{\partial \boldsymbol{z}(\boldsymbol{y}, t)}{\partial \boldsymbol{y}}}_{\boldsymbol{C}_{\mathrm{T}}} \boldsymbol{v} - \dot{\boldsymbol{z}}_{\mathrm{des}} = \boldsymbol{C}_{\mathrm{T}} \boldsymbol{v} - \dot{\boldsymbol{z}}_{\mathrm{des}} = \boldsymbol{0}. \tag{5.29}$$

Here $\boldsymbol{C}_{\mathrm{T}}$ is the translational Jacobian of the system output $\boldsymbol{z}$ of the forward model (5.20). The inverse model of the test system from Sec. 3.1 can then be written as index 1 DAE in the form

$$\boldsymbol{v} = \dot{\boldsymbol{y}} \tag{5.30}$$

$$\boldsymbol{M}(\boldsymbol{y}, t)\dot{\boldsymbol{v}} = \boldsymbol{q}(\boldsymbol{y}, \boldsymbol{v}, t) - \boldsymbol{k}(\boldsymbol{y}, \boldsymbol{v}, t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{\Phi}(\boldsymbol{F}_{\mathrm{red}})\boldsymbol{F}_{\mathrm{red}}(t) \tag{5.31}$$

$$\boldsymbol{C}_{\mathrm{T}}\boldsymbol{v} = \dot{\boldsymbol{z}}_{\mathrm{des}}. \tag{5.32}$$

In matrix notation this yields

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M} & -\boldsymbol{B}\boldsymbol{\Phi} \\ \boldsymbol{C}_{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{y}} \\ \dot{\boldsymbol{v}} \\ \boldsymbol{F}_{\mathrm{red}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{q} - \boldsymbol{k} \\ \dot{\boldsymbol{z}}_{\mathbf{des}} \end{bmatrix}. \tag{5.33}$$

With the chosen discretization of the PCC model into $N_{\mathrm{pieces}} = 1$ pieces, a differentially flat system is obtained, corresponding to a fully actuated system. The system has no internal dynamics. For finer discretizations, flatness is lost because the number of independent generalized coordinates is greater than the number of servo constraint equations, and internal dynamics remain.

In the following, this system is solved using the ode15s solver from MATLAB, which is based on the trapezoidal rule. Other solvers can also be used. In [DrückerSeifried23] an overview of the performance of different solvers for feed-forward control using the servo-constraints approach is given. The solution of the

DAE (5.30) to (5.32) gives the generalized position coordinates $\boldsymbol{y}$, generalized velocity coordinates $\boldsymbol{v}$ and the linear impulse $\boldsymbol{J} = \int \boldsymbol{F}_{\mathrm{red}} \mathrm{d}t$, which is the integral of the reduced tendon forces $\boldsymbol{F}_{\mathrm{red}}$ with respect to time.

In a post-processing step, the control signals for the servos can be calculated from these values. First, the reduced tendon forces $\boldsymbol{F}_{\mathrm{red}}$ are determined by numerically differentiating the linear impulse $\boldsymbol{J} = \int \boldsymbol{F}_{\mathrm{red}} \mathrm{d}t$ with respect to time. The tendon forces $\boldsymbol{F}$ are then calculated from the reduced tendon forces $\boldsymbol{F}_{\mathrm{red}}$ using

$$\boldsymbol{F} = \boldsymbol{B}(\boldsymbol{y}) \boldsymbol{\Phi}^{\dagger} \boldsymbol{F}_{\mathrm{red}}(t). \tag{5.34}$$

Finally, the tendon length $\boldsymbol{s}$ is calculated as described in Sec. 2.6.2 resulting in

$$\boldsymbol{s} = \boldsymbol{s}_{\mathrm{geo}}(\boldsymbol{y}) + \boldsymbol{C}_{\mathrm{tendon}}^{-1} \boldsymbol{F}, \tag{5.35}$$

where $\boldsymbol{C}_{\mathrm{tendon}}$ introduced in equation (2.95) is the tendon stiffness matrix which has to be determined experimentally and $\boldsymbol{s}_{\mathrm{geo}}(\boldsymbol{y})$ is the change in tendon length due to the change in geometry of the soft robot during deformation. For the soft robotic test system considered here the tendon stiffness matrix is $\boldsymbol{C}_{\mathrm{tendon}} = \mathrm{diag}\,(33\,\mathrm{N/m},\ 33\,\mathrm{N/m},\ 33\,\mathrm{N/m})$. The tendon length $\boldsymbol{s}$ can now be used directly to control the soft robot. The resulting controller structure is shown in Fig. 5.8.

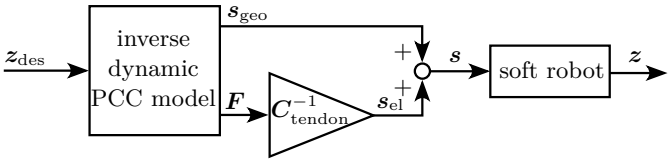


Figure 5.8: Structure of the inverse dynamic controller.

5.3 Hybrid Data-Driven Quasi-Static and Model-Based Dynamic Model

For many soft robots, as well as for the test system considered in this work, it can be observed that the quasi-static behavior is strongly affected by parameter uncertainties, while the dynamic behavior is less affected. In addition, data-driven models are usually very efficient and accurate for the quasi-static modeling of soft robots, while the dynamic data-driven modeling is usually less efficient and accurate because it is difficult to collect a sufficient amount of training data. Therefore, it is useful to combine a data-driven quasi-static model with a model-based dynamic model. In the following, the quasi-static data-driven controller presented in Sec. 5.1 and the dynamic model-based controller presented in Sec. 5.2 are combined into a hybrid controller. The structure of the hybrid controller is shown in Fig. 5.9. For the hybrid controller, the quasi-static behavior is modeled by the data-driven quasi-static model and the dynamic behavior is modeled by the model-based dynamic model.

For the combination of the two controllers, first the composition of the tendon length $\boldsymbol{s}$ has to be considered. As described in Sec. 2.6, the tendon length can be divided into a geometric part $\boldsymbol{s}_{\mathrm{geo}}$ and an elastic part $\boldsymbol{s}_{\mathrm{el}}$. The geometric part $\boldsymbol{s}_{\mathrm{geo}}$ results from the change in geometry of the soft robot during deformation. The elastic part $\boldsymbol{s}_{\mathrm{el}}$ results from the elongation of the tendons due to their elasticity. The elastic part $\boldsymbol{s}_{\mathrm{el}}$ can be further divided into a part $\boldsymbol{s}_{\mathrm{stiff}}$ resulting from the stiffness of the soft robot and a part $\boldsymbol{s}_{\mathrm{dyn}}$ resulting from other dynamic forces $\boldsymbol{F}_{\mathrm{dyn}}$ such as damping and inertia. This can be written as

$$\boldsymbol{s} = \boldsymbol{s}_{\mathrm{geo}} + \underbrace{\boldsymbol{s}_{\mathrm{stiff}} + \boldsymbol{s}_{\mathrm{dyn}}}_{\boldsymbol{s}_{\mathrm{el}}} . \tag{5.36}$$

For the quasi-static behavior of the soft robot, the geometric part $\boldsymbol{s}_{\mathrm{geo}}$ and the stiffness part $\boldsymbol{s}_{\mathrm{stiff}}$ of the tendon force are relevant. For the dynamic behavior, the dynamic part $\boldsymbol{s}_{\mathrm{dyn}}$ is also important. The tendon length $\boldsymbol{s}_{\mathrm{NN}}$ calculated by

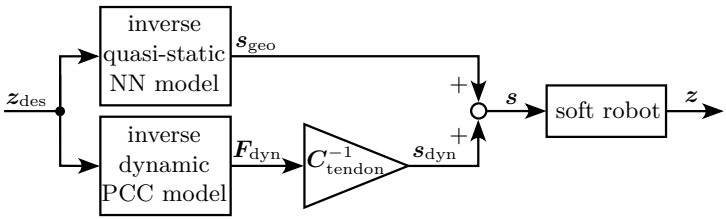


Figure 5.9: Structure of the hybrid controller.

the quasi-static controller can be written as

$$\boldsymbol{s}_{\mathrm{NN}} = \boldsymbol{s}_{\mathrm{geo}} + \boldsymbol{s}_{\mathrm{stiff}} + \boldsymbol{s}_{\mathrm{err}}. \tag{5.37}$$

It includes the tendon length due to the geometry and stiffness of the soft robot, as well as a learned approximation $\boldsymbol{s}_{\mathrm{err}}$ of unmodeled effects. Dynamic effects from damping and inertia are neglected.

The tendon length calculated by the inverse dynamics controller can be written as

$$\boldsymbol{s}_{\mathrm{inv}} = \boldsymbol{s}_{\mathrm{geo}} + \underbrace{\boldsymbol{s}_{\mathrm{stiff}} + \boldsymbol{s}_{\mathrm{dyn}}}_{\boldsymbol{s}_{\mathrm{el}} = \boldsymbol{C}_{\mathrm{tendon}}^{-1}\boldsymbol{F}}. \tag{5.38}$$

This includes all modeled effects on the tendon length but does not include unmodeled effects. In a post-processing step, the dynamic part $\boldsymbol{s}_{\mathrm{dyn}}$ can be calculated to

$$\boldsymbol{s}_{\mathrm{dyn}} = \boldsymbol{C}_{\mathrm{tendon}}^{-1}\boldsymbol{F}_{\mathrm{dyn}} \tag{5.39}$$

where $\boldsymbol{F}_{\mathrm{dyn}}$ is the part of the tendon forces $\boldsymbol{F}$ resulting from the dynamics of the system. To calculate $\boldsymbol{F}_{\mathrm{dyn}}$, first the generalized accelerations $\dot{\boldsymbol{v}}$ are calculated by numerically differentiating the generalized velocities $\boldsymbol{v}$ with respect to time. Then the tendon forces $\boldsymbol{F}_{\mathrm{red,dyn}}$ resulting from the dynamics are computed by inserting the generalized positions $\boldsymbol{y}$, velocities $\boldsymbol{v}$ and accelerations $\dot{\boldsymbol{v}}$ computed by the inverse model into equation (5.31), setting the stiffness of the soft robot to zero and solving this linear system of equations for the reduced tendon forces $\boldsymbol{F}_{\mathrm{red}}$. In a next step, the tendon forces $\boldsymbol{F}_{\mathrm{dyn}}$ are calculated from the reduced tendon forces $\boldsymbol{F}_{\mathrm{red,dyn}}$ using equation (5.34). Then, the dynamic tendon length $\boldsymbol{s}_{\mathrm{dyn}}$ is calculated from the tendon forces $\boldsymbol{F}_{\mathrm{dyn}}$ using equation (5.39).

Finally, the outputs of the two controllers can simply be summed to get the tendon length $\boldsymbol{s}$ of the combined controller. The tendon length $\boldsymbol{s}$ can now be used directly to control the soft robotic test system.

5.4 Experimental Evaluation

In the following, the trajectory tracking control performance of the three feedforward controllers presented in Secs. 5.1 to 5.3 is investigated and compared regarding their accuracy and computational efficiency.

5.4.1 Trajectory

In a first step, the trajectory tracking control of a triangular trajectory with rounded corners and an edge length of ≈140 mm is considered. The triangular trajectory $\boldsymbol{z}_{\mathrm{des}}(t)$ is defined as

$$\boldsymbol{z}_{\mathrm{des}}(t) = \begin{bmatrix} b_1 \cos(w(t) \cdot 2\pi T) + b_2 \cos(2w(t) \cdot 2\pi T) \\ -b_1 \sin(w(t) \cdot 2\pi T) + b_2 \sin(2w(t) \cdot 2\pi T) \end{bmatrix} \tag{5.40}$$

where $b_1 = 70\,\mathrm{mm}$, $b_2 = b_1/4$, T is the period time for one full turn and $w(t)$ is the curve parameter. The trajectory is twice continuously differentiable. It starts at the most right point of the triangle with zero velocity. In a first lap the velocity is slowly increased within a time of $2T$ until the desired velocity is reached. Then follow three laps with constant velocity and period time T. In a last lap the velocity is reduced to zero again within a time of $2T$. The curve parameter w is defined as a twice continuously differentiable polynomial of fifth order with $w(0) = 0$, $w(7T) = 5$, $\dot{w}(0) = \dot{w}(7T) = 0$ and $\ddot{w}(0) = \ddot{w}(7T) = 0$ at start and end time.

5.4.2 Trajectory Tracking Results

To evaluate the performance of the controller, the tracking of the triangular trajectory with both a period time of $T = 5\,\mathrm{s}$ and a period time of $T = 1\,\mathrm{s}$ is considered. The triangular trajectory with a period time of $T = 5\,\mathrm{s}$ represents a motion that is just slow enough to be treated as quasi-static. The triangular trajectory with a period time of $T = 1\,\mathrm{s}$ represents a highly dynamic trajectory, which is at the upper limit of what can be realized with the test system used. The main limitation of the test system is the servos, which are not fast and powerful enough for even more agile trajectories.

For the evaluation of the trajectory tracking control performance the control error $e(t)$ defined as

$$e(t) = ||\boldsymbol{z}_{\mathrm{des}}(t) - \boldsymbol{z}_{\mathrm{exp}}(t)|| \tag{5.41}$$

is considered. Here, $\boldsymbol{z}_{\mathrm{exp}}$ is the measured tip position obtained from the experiment.

Slow Trajectory

First, the slow trajectory with a period time of $T = 5\,\mathrm{s}$ is considered. From Fig. 5.10 it can be seen that all three controllers are able to follow the trajectory accurately. The hybrid controller performs best with a mean error of $e_{\mathrm{mean}} = 2.9\,\mathrm{mm}$ and a maximum error of $e_{\mathrm{max}} = 8.9\,\mathrm{mm}$. The NN controller performs slightly worse, but still achieves very accurate trajectory tracking with a

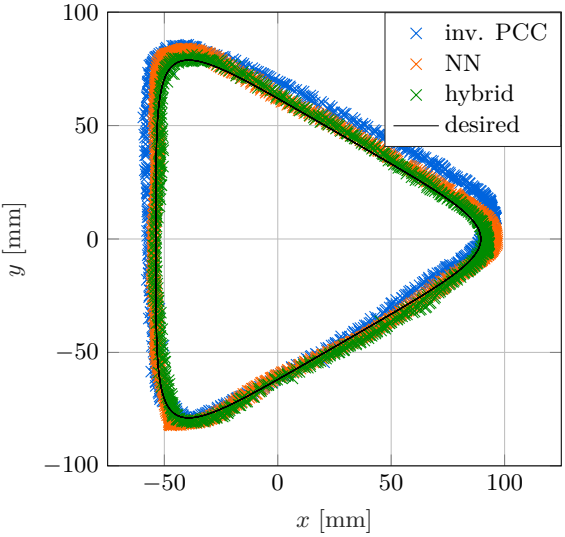


Figure 5.10: Trajectory tracking results of the slow triangular trajectory with a period time of $T = 5\,\mathrm{s}$ for the inverse dynamic PCC controller and inverse quasi-static NN controller.

mean error of $e_{\mathrm{mean}} = 4.5\,\mathrm{mm}$ and a maximum error of $e_{\mathrm{max}} = 13.0\,\mathrm{mm}$. This can be explained by the fact that the NN controller has the same accurate quasi-static model as the hybrid controller, but does not take into account the dynamics. For this reason, the controller performs slightly worse in the corners, which are the most dynamic parts of the trajectory. For even slower trajectories, the difference in behavior between the NN controller and the hybrid controller disappears. The dynamic PCC controller also allows for accurate trajectory tracking. However, with a mean error of $e_{\mathrm{mean}} = 12.5\,\mathrm{mm}$ and a maximum error of $e_{\mathrm{max}} = 42.5\,\mathrm{mm}$, it has four times the control error of the hybrid controller. In particular, it can be seen that the trajectory tracked by the PCC controller is slightly rotated towards the desired trajectory. This can be explained by the less accurate representation of the quasi-static behavior compared to the other two controllers, which use a data-driven approach to represent the quasi-static behavior. Thus, they include effects such as manufacturing inaccuracies that are not included in the PCC model used by the PCC controller. Note that even though the PCC controller includes the dynamics of the system while the quasi-static NN controller does not, it still performs worse because for the slow trajectory considered here, the influence of the quasi-static behavior is much larger than the influence of damping and inertia. In detail, this can be seen from the two controllers based on the servo-constraints approach, which provide insight into the contribution of the

forces. For the inverse dynamic PCC controller, in the mean, the tendon forces resulting from inertia and damping are only 1.6 % of the mean of the total tendon forces while the rest accounts for the stiffness part.

Fast Trajectory

Second, the fast trajectory with a period time of $T = 1\,\mathrm{s}$ is considered. It can be clearly seen from Fig. 5.11 that the quasi-static NN controller is not able to follow the trajectory. The resulting trajectory is much too large, rotated against the desired trajectory, and contains some extra loops in the corners. This can be explained by the fact that the controller neglects inertia and damping, which play an important role for the fast trajectory. Especially the centrifugal forces are relevant. The influence of the dynamics is examined in more detail in Sec. 5.4.3. In contrast, it can be seen from Fig. 5.12 that both dynamic controllers are able to follow the trajectory. With the inverse dynamic PCC controller, a mean error of $e_{\mathrm{mean}} = 19.7\,\mathrm{mm}$ and a maximum error of $e_{\mathrm{max}} = 49.5\,\mathrm{mm}$ could be achieved, which is a reasonable performance for a soft robot. For the hybrid controller, the control error is less than have as large with a mean error of $e_{\mathrm{mean}} = 8.6\,\mathrm{mm}$ and a maximum error of $e_{\mathrm{max}} = 18.4\,\mathrm{mm}$. This is a very good performance for a fast-moving soft robot. The better performance of the hybrid controller compared to the inverse-dynamic PCC controller can be explained by the more accurate representation of the quasi-static behavior of the soft robot.

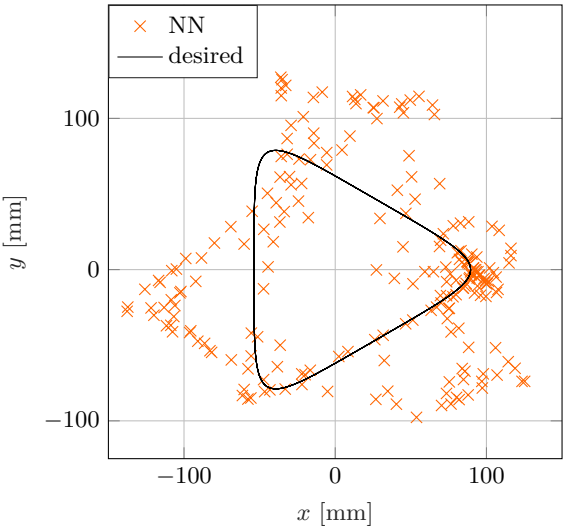


Figure 5.11: Trajectory tracking results of the fast triangular trajectory with a period time of $T = 1\,\mathrm{s}$ for the inverse quasi-static NN controller.

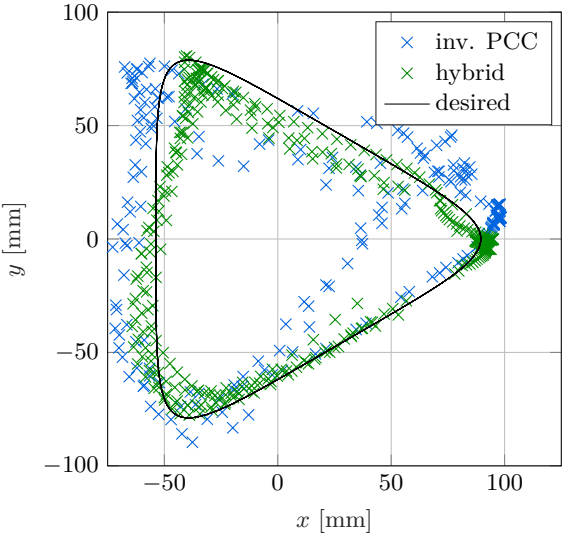


Figure 5.12: Trajectory tracking results of the fast triangular trajectory with a period time of $T = 1\,\mathrm{s}$ for the inverse dynamic PCC controller and hybrid inverse dynamic controller.

Besides the control error, the system input is also of interest. In Fig. 5.13, the contribution of the quasi-static NN part and the inverse-dynamic part to the control output of the hybrid controller is shown for the first tendon as an example. The results for the other two tendons are qualitatively comparable and are therefore omitted here. It can be seen that the inverse dynamic part of the controller compensates for the centrifugal forces, resulting in a lower amplitude of the tendon length calculated by the hybrid controller compared to the quasi-static NN controller. This is discussed in more detail in Sec. 5.4.3.

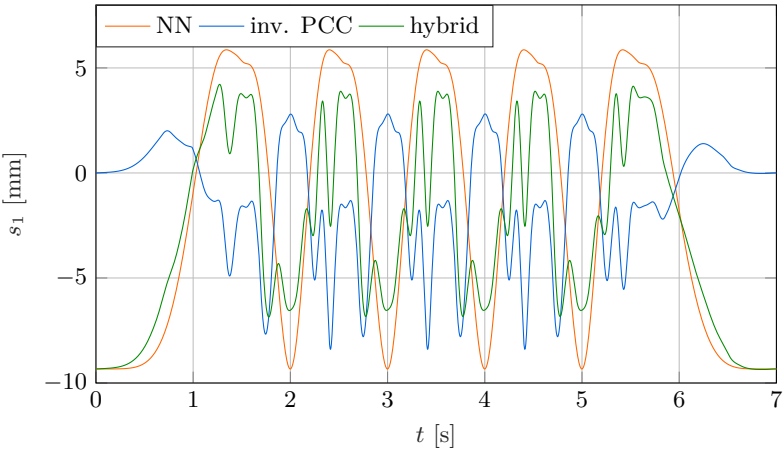


Figure 5.13: Contribution of the quasi-static NN part and the inverse-dynamic part of the hybrid controller to the tendon length change s_1 of the first tendon for the fast triangular trajectory with $T = 1\,\mathrm{s}$.

Comparison of the Control Performance for the two Trajectory

In Tabs. 5.2 and 5.3 the mean and maximum control errors are listed for the two trajectories and the three controllers. It can be seen that for both the slow and the fast trajectory, the hybrid controller performs best. This can be explained by the fact that it is based on the most accurate representation of the quasi-static and dynamic behavior. For the slow trajectory, the second best controller is the NN quasi-static controller and the worst controller is the PCC dynamic controller. For the fast trajectory, it is the other way around. This can be explained by the fact that the NN quasi-static controller has the more accurate representation of the quasi-static behavior, while the PCC dynamic controller has the better representation of the inertia and damping. The quasi-static behavior is more important for the slow trajectory, while inertia and damping dominate for the fast trajectory. Note, however, that while all three controllers achieve good performance on the slow trajectory, only the PCC dynamic controller and the hybrid controller are able to follow the dynamic trajectory, while the NN quasi-static controller fails.

Table 5.2: Mean and maximum control error for the slow trajectory with a period time of 5 s and different quasi-static and dynamic controllers.

	NN quasi-static	**PCC dynamic**	**hybrid**
e_{mean}	4.5 mm	12.5 mm	2.9 mm
e_{max}	13.0 mm	42.3 mm	8.9 mm

Table 5.3: Control error for the fast trajectory with a period time of 1 s and different quasi-static and dynamic controllers.

	NN quasi-static	**PCC dynamic**	**hybrid**
e_{mean}	35.9 mm	19.7 mm	8.6 mm
e_{max}	90.3 mm	49.5 mm	18.4 mm

Computational Efficiency

On a computer with an "Intel Core i7-1280P" CPU, computing the system inputs for both the inverse dynamic PCC controller and the hybrid controller takes 9.45 s for the slow trajectory and 4.31 s for the fast trajectory. As the slow trajectory has a total duration of 45 s and the fast trajectory has a total duration of 17 s, this is clearly faster than real-time. Note, however, that real-time capability is not guaranteed because the `ode15s` controller uses step width control. The computation time for the inverse quasi-static NN controller is independent of the choice of trajectory. The computation takes 115 μs for a trajectory with a duration of 1 s, assuming a sample rate of 50 Hz, which is the sample rate at which the servos operate. This shows that the quasi-static NN controller is clearly real-time capable.

5.4.3 Analysis of the Influence of the Dynamics

In a second step, the influence of the dynamics of the soft robot on the tendon forces required for trajectory tracking is investigated in more detail in order to gain a better understanding of the controller behavior. For this purpose, the trajectory tracking control of a circle with a radius of $r = 90$ mm and different constant angular velocities $\omega = 0.015\,1/\text{s} \ldots 63\,1/\text{s}$, corresponding to period times of $T = 420\,\text{s} \ldots 0.1\,\text{s}$, is considered in simulation. For control the PCC inverse dynamics controller presented in Sec. 5.2 is used. Note that the maximum angular velocity that can be realized experimentally on the test system is only $\omega_{\text{exp,max}} \approx 2\pi 1/\text{s}$, since for higher angular velocities the actuator limits are exceeded. The circular trajectory is chosen because it provides a good insight into the dynamic behavior of the test system.

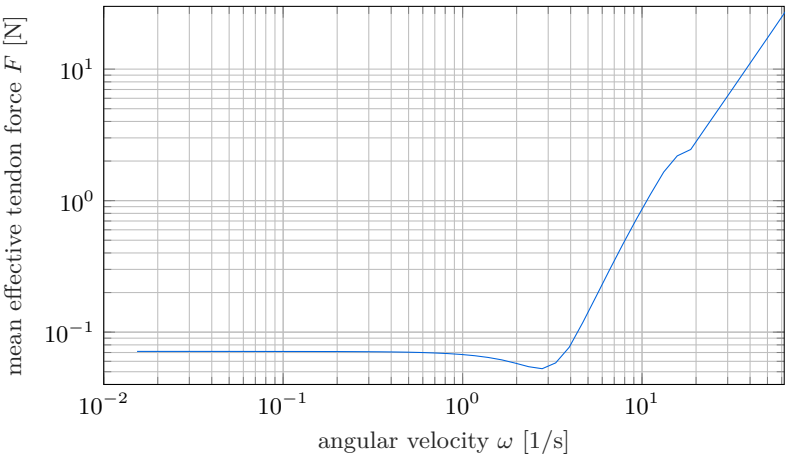
In the following, the effective tendon force $\boldsymbol{F}_{\text{eff}}$, which is motivated by the reduced tendon force $\boldsymbol{F}_{\text{red}}$ introduced in Sec. 3.1.2 and defined as

$$\boldsymbol{F}_{\text{eff}} = F_1 \boldsymbol{R}_z(0^\circ)\boldsymbol{e}_x + F_2 \boldsymbol{R}_z(120^\circ)\boldsymbol{e}_x + F_3 \boldsymbol{R}_z(240^\circ)\boldsymbol{e}_x \tag{5.42}$$

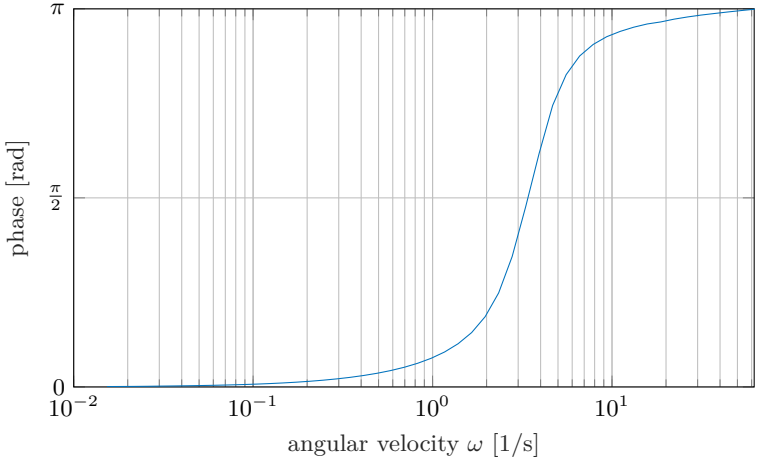
is considered. Here $\boldsymbol{R}_z(\alpha)$ is the rotation matrix around the $\boldsymbol{z}$-axis with the angle α and $\boldsymbol{e}_x$ is the unit vector in x-direction. Both, the effective tendon force and the tip position are considered in polar coordinates.

In Fig. 5.14a the mean magnitude F of the effective tendon force $\boldsymbol{F}_{\text{eff}}$ is plotted over the angular velocity. It can be seen that for slow motions with an angular velocity of up to $\omega = 0.6\,1/\text{s}$ there is almost no dependence of the tendon force on the angular velocity. For faster motions the tendon force decreases and reaches its minimum at an angular velocity of $\omega = 2.8\,1/\text{s}$ and starts to increase quadratically with increasing angular velocity. Additionally, in Fig. 5.14b the phase difference between the tip position $\boldsymbol{z}$ and the effective tendon force $\boldsymbol{F}_{\text{eff}}$ is shown. It can be seen that at low angular velocities the effective tendon force and the tip position have the same phase, while at higher angular velocities the phase difference approaches π.

This behavior can be explained by the increasing influence of inertia and damping forces with increasing angular velocity, while the contribution of stiffness forces is independent of angular velocity. Therefore, the stiffness forces dominate at low angular velocities. Since the stiffness forces act directly against a deformation of the soft robot, the effective tendon forces point in the direction of deflection of the soft robot. As the angular velocity increases, the damping forces and the inertial forces increase. In particular, the centripetal forces, which increase quadratically with angular velocity, dominate at high angular velocities. Note that the centripetal forces point in the opposite direction to the stiffness forces, which causes the phase difference between the tendon forces and the tip position to approach π as the centripetal forces begin to dominate. At an angular velocity of $\omega \approx 2.8\,1/\text{s}$, the centripetal forces compensate for the stiffness forces, so that the tendon forces only need to compensate for the damping forces, resulting in minimal tendon forces.



a) Mean tendon force.



b) Mean phase shift between effective tendon forces and tip position.

Figure 5.14: Mean tendon force and phase for a circular trajectory and different angular velocities ω.

Feedback Control

As soft robotic applications evolve, there is a growing demand for more sophisticated control mechanisms, with an emphasis on agility and precision. While feedforward controllers address the issue that sensor integration into soft robots is usually difficult, parameter uncertainties and external disturbances demand for feedback control. In soft robotics, besides external camera tracking systems, integrated curvature sensors as e.g. presented in chapter 4 can be used.

In soft robotics, various proportional (P), proportional-derivative (PD), proportional-integral (PI), and proportional-internal-derivative (PID) controllers are used. These are often combined with feedback linearization [KapadiaWalker11, FalkenhahnEtAl15, FalkenhahnEtAl17], inverse physical models [Della SantinaEtAl20b], or inverse data-driven models [LeeEtAl17b, YuEtAl19]. To account for the strong nonlinearities of soft robots, adaptive controllers [MelinguiEtAl18, TrumicEtAl21, ChenEtAl24a] are also used. Other control concepts such as sliding mode control [KapadiaEtAl10] or energy shaping control based on a port-Hamiltonian model [Franco22] are also applied.

Another very popular class of controllers are optimal control and model predictive control approaches based on mechanical models [BestEtAl16], data-driven models [GillespieEtAl18] or online estimation of the Jacobian [YipCamarillo14, YipCamarillo16, OuyangEtAl18]. Especially when the derivation of models is difficult, purely data-driven control approaches are also used. For slow motion, direct learning of the controller based on feedforward neural networks [ThuruthelEtAl17a] or locally weighted projection regression (LWPR) [TangEtAl22] is used. For more complex control problems, various quasi-static [EngelEtAl05, SilverEtAl14, ZhangEtAl17] and dynamic [ThuruthelEtAl19, LiEtAl22] reinforcement learning approaches are applied.

In this chapter some feedback control concepts for soft robots are investigated and compared. In the first part of this chapter, in Sec. 6.1 the application of simple feedback controllers to a physical test system with integrated curvature sensors is investigated in experiments. In the second part of this chapter more advanced feedback control concepts for soft robots are investigated and compared by simulation. In a first step, in Sec. 6.2 a test system used to evaluate the controller performance in simulation is presented. Then, in Secs. 6.3 to 6.5, a data-driven quasi-static controller, a linear quadratic regulator with gain scheduling,

and a model predictive controller are presented. Finally, in Sec. 6.6 the trajectory tracking control performance of the three controllers is compared in terms of accuracy, robustness to parameter uncertainty, and computational efficiency. This work has also been published in [GrubeEtAl22, GrubeEtAl25].

6.1 Basic Combined Feedforward and Feedback Control

In the following, the application of a simple combined feedforward and feedback controller for trajectory tracking control is examined in experiment. As a test system, the beam-shaped soft robotic test system introduced in Sec. 3.1 is considered, with the spatial curvature sensor introduced in Sec. 4.2 integrated into the soft robot. In a first step the controller structure is presented. Then, in a second step, the performance of the trajectory tracking controller is evaluated experimentally.

6.1.1 Controller Structure

The controller structure is shown in Fig. 6.1. The inputs of the controller are the desired tip position $\boldsymbol{z}_{\mathrm{des}} \in \mathbb{R}^{2\times 1}$ and the current tip position $\boldsymbol{z}_{\mathrm{curv}} \in \mathbb{R}^{2\times 1}$ measured by the curvature sensor. The outputs of the controller are the tendon length $\boldsymbol{s} \in \mathbb{R}^{3\times 1}$.

As a feedforward controller, the neural network based quasi-static controller presented in Sec. 5.1 is used. The input of the feedforward controller is the sum

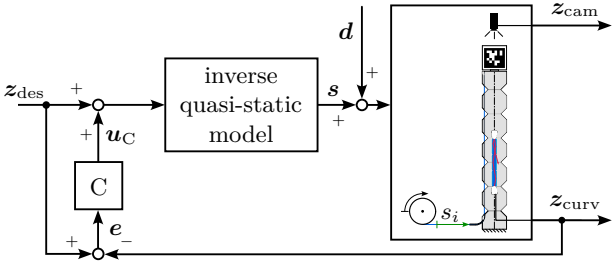


Figure 6.1: Control circuit with output distortion d.

of the desired tip position $\boldsymbol{z}_{\mathrm{des}} \in \mathbb{R}^{2\times 1}$ and the output $\boldsymbol{u}_{\mathrm{C}} \in \mathbb{R}^{2\times 1}$ of the feedback part of the controller described below. The output of the feedforward controller is the tendon length vector $\boldsymbol{s} \in \mathbb{R}^{3\times 1}$. The redundancy of the soft robotic test system is completely resolved by the feedforward part of the controller. This is described in detail in Sec. 3.1.2. The training data collected in Sec. 4.2.2 is used to train the neural network of this controller.

A proportional (P) and a proportional-integral (PI) controller are compared as the feedback part of the controller. The feedback controller is 2×2 in size. Since the soft robot is symmetric, a diagonal proportional gain matrix $\boldsymbol{K}_{\mathrm{P}} = \mathrm{diag}(K_{\mathrm{P}}, K_{\mathrm{P}})$ and a diagonal integral gain matrix $\boldsymbol{K}_{\mathrm{I}} = \mathrm{diag}(K_{\mathrm{I}}, K_{\mathrm{I}})$ with identical gains in x- and y-direction are chosen. The input of the feedback controller is the tip position error $\boldsymbol{e} \in \mathbb{R}^{2\times 1}$ which is defined as

$$\boldsymbol{e} = \boldsymbol{z}_{\mathrm{des}} - \boldsymbol{z}_{\mathrm{curv}}. \tag{6.1}$$

The output $\boldsymbol{u}_{\mathrm{C}} \in \mathbb{R}^{2\times 1}$ of the feedback controller is calculated as

$$\boldsymbol{u}_{\mathrm{C}} = \boldsymbol{K}_{\mathrm{P}}\boldsymbol{e} \tag{6.2}$$

for the P controller and as

$$\boldsymbol{u}_{\mathrm{C}} = \boldsymbol{K}_{\mathrm{P}}\boldsymbol{e} + \boldsymbol{K}_{\mathrm{I}} \int \boldsymbol{e}\mathrm{d}t \tag{6.3}$$

for the PI controller. The integral $\int \boldsymbol{e}\mathrm{d}t$ of the control error required for the PI controller is calculated numerically. The output of the feedback controller is added to the desired tip position $\boldsymbol{z}_{\mathrm{des}}$ and fed into the feedforward controller. In the following, the gains $K_{\mathrm{P}} = 0.2$ and $K_{\mathrm{I}} = 1$ are used. For larger gains of $K_{\mathrm{P}} = 0.3$ and $k_{\mathrm{I}} = 2$ the system starts to oscillate before it becomes unstable for even larger gains. In addition to the combined feedforward and feedback controllers, the pure quasi-static feedforward controller, i.e. $\boldsymbol{u}_{\mathrm{C}}$, is also considered as a reference in the following.

6.1.2 Experimental Evaluation of the Controller Performance

In the following, the performance of the trajectory tracking control is examined on a circular trajectory with a radius of 120 mm. The circular trajectory starts at the center of the circle. Then, the soft robot slowly moves to the outside of the circle within 10 s. After that, the soft robot makes a complete counterclockwise rotation around the circle at a constant speed of 50 mm/s and finally returns to the center. There are two types of distortions considered: The first type of distortion is a weight attached to the tip of the soft robot as shown in Fig. 6.2. This weight causes an eccentric force at the tip of the soft robot due to gravity and thus an additional deflection of the soft robot depending on the tip position.

The second type of distortion is an artificial constant offset $\boldsymbol{d} = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^\top$ mm which affects the system input signals $\boldsymbol{s}$ as shown in Fig. 6.1. This results in an offset of 10 % from the maximum value at the first tendon.

The results of the trajectory tracking control for both types of distortions are shown in Figs. 6.3 and 6.4. It can be seen that for both types of distortions, the pure feedforward controller gives the worst results with large deviations from the desired trajectory. A root mean square error of 39.2 mm was obtained for the distortion with the added weight and 31.3 mm for the output distortion $\boldsymbol{d}$. This can be explained by the fact that the feedforward controller does not take into account any measurement data and therefore does not know of the distortions.

With the addition of the P controller, the control error is slightly better, with an error of 30.6 mm for the distortion with the weight and 26.9 mm for the output distortion $\boldsymbol{d}$. However, the trajectory tracking error is still large. This is due to the relatively small gain of the P controller, which had to be chosen for stability reasons.

With the addition of the PI controller, very good trajectory tracking results can be achieved with a control error of only 7.9 mm for the weight distortion and 3.1 mm for the output distortion $\boldsymbol{d}$. For the output distortion, the control error disappears almost completely. The remaining error in this case can mainly be explained by tendons becoming loose due to the distortion, which makes the control more difficult. For the distortion with the added weight at the tip of the robot, a small control error remains, which depends directly on the chosen gain of the controller. Here, too, stability considerations limit the maximum gain. It can be concluded that overall accurate trajectory tracking control could be achieved, for more agile applications, however, more advanced control concepts are needed.

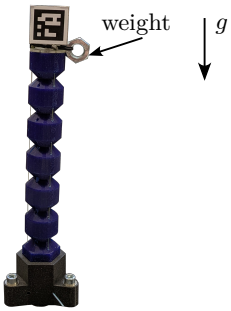


Figure 6.2: Distortion of the soft robot by additional weight attached to the tip.

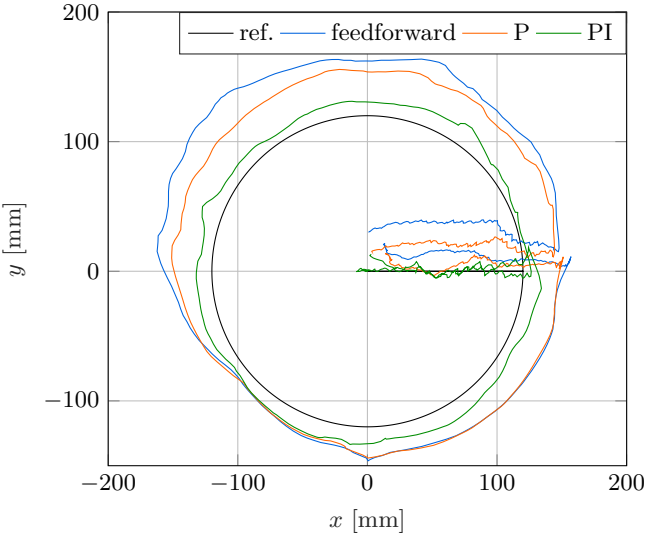


Figure 6.3: Circular trajectory with distortion with weight for different controllers.

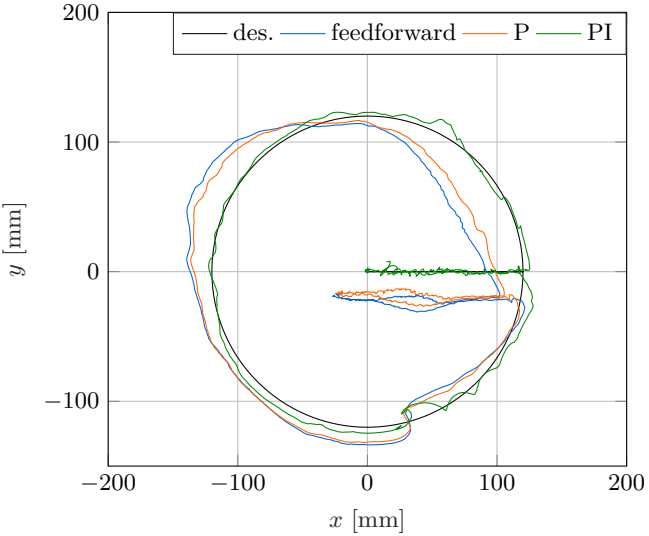


Figure 6.4: Circular trajectory with output distortion $\boldsymbol{d}$ for different controllers.

6.2 Test System for Testing Advanced Controllers

In the following, a simple 2D tendon-actuated soft robot arm is considered as a test system in simulation for the comparison of the controllers. First the design of the test system is presented. Then the equations of motion of the test system are derived.

6.2.1 Design of the Soft Robotic Test System

As a test system, a simple tendon-actuated soft robot arm as shown in Fig. 6.5 is considered. The soft robot has a slender shape, and its cross-section forms a circular ring with an outer radius of $R = 50\,\text{mm}$ and an inner radius of $r = 25\,\text{mm}$. It has a total length of $L_{\text{total}} = 500\,\text{mm}$ and a Young's modulus of $E = 7.32 \cdot 10^5\,\text{N/m}^2$. Stiffness proportional damping is assumed with a damping constant of $E\mu = 36.6 \cdot 10^3\,\text{Nms}$. The soft robot is actuated by three pairs of tendons that run along the top and bottom of the soft robot as shown in Fig. 6.5. The first tendon pair ends at 1/3 of the length of the soft robot, the second tendon pair ends at 2/3 of the length of the soft robot and the third tendon pair runs to the end of the soft robot. The tendons have a distance of $r_{\text{tendon}} = 25\,\text{mm}$ to the neutral fiber. The tendon forces of the three tendon pairs are considered as system input. The tip position $\boldsymbol{z} \in \mathbb{R}^{2\times 1}$ is considered as system output. The geometric and material properties are summarized in Tab. 6.1. For simplicity, only movements in the xz-plane are considered and modeled.

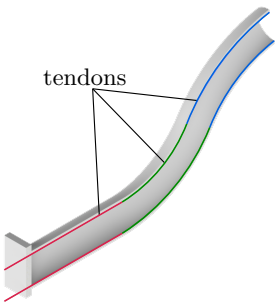


Figure 6.5: Cut through the soft robot.

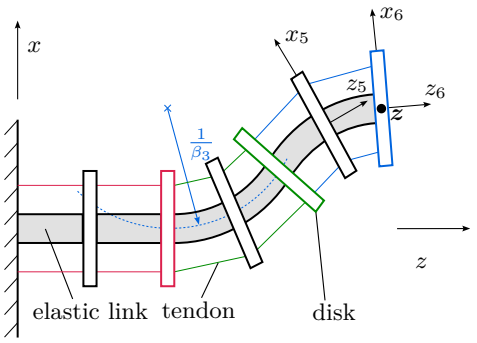


Figure 6.6: Schematic representation of the 2D system with 6 segments described as the PCC model.

Table 6.1: Parameters of the simulation model.

Variable	Description	Value
L_{total}	total length	500 mm
r	inner radius	25 mm
R	outer radius	50 mm
r_{tendon}	distance of cables to centerline	25 mm
E	Young's modulus	$7.32 \cdot 10^5$ N/m^2
$E\mu$	damping constant	$36.6 \cdot 10^3$ Ns/m^2

6.2.2 Modeling

The soft robot is modeled with the 2D version of the PCC model described in Sec. 2.1. A discretization into 6 pieces is used. As the soft robot is very stiff in longitudinal direction, elongation is neglected. The deformation of the soft robot can be fully described by the curvatures β_i of the individual pieces. The discretized soft robot is shown schematically in Fig. 6.6.

Following Sec. 2.1 for the derivation of the equations of motion of the PCC model and Sec. 2.6.2 for the integration of the tendon actuation, the equations of motion for the 2D test system can be written in the form

$$\boldsymbol{M}(\boldsymbol{y},t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) = \boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{F}(t), \tag{6.4}$$

where $\boldsymbol{y} = \begin{bmatrix}\beta_1 & \dots & \beta_6\end{bmatrix}^\top$ are the generalized coordinates, $\boldsymbol{F} = \begin{bmatrix}F_{1,\text{up}} & F_{1,\text{low}} & F_{2,\text{up}} & F_{2,\text{low}} & F_{3,\text{up}} & F_{3,\text{low}}\end{bmatrix}^\top$ is the vector of tendon forces, $\boldsymbol{M}(\boldsymbol{y},t) \in \mathbb{R}^{6\times 6}$ is the mass matrix, $\boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) \in \mathbb{R}^{6\times 1}$ is the vector of generalized Coriolis, centrifugal, and gyroscopic forces, $\boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t) \in \mathbb{R}^{6\times 1}$ is the vector of generalized applied forces and moments and $\boldsymbol{B} \in \mathbb{R}^{6\times 6}$ is the input distribution matrix. Here, $F_{k,\text{up}}$ and $F_{k,\text{low}}$ are the tendon forces of the upper and the lower tendon of tendon pair k where $k = 1 \dots 3$ is the tendon pair index. Note that as tendons can only transmit pulling forces the tendon forces always have to be positive.

Considering the tip position $\boldsymbol{z} \in \mathbb{R}^{2\times 1}$ as system output and the tendon forces $\boldsymbol{F} \in \mathbb{R}^{6\times 1}$ as system input, the system is redundantly actuated. Two different sources of redundancy contribute to the redundancy of this system: First, there is kinematic redundancy, which results from the fact that most points in the workspace can be reached with more than one configuration of the soft robot. Second, there is an actuation redundancy. This results from the fact that for the actuation of the system only the difference $F_{k,\text{up}} - F_{k,\text{down}}$ of the tendon forces in each tendon pair is relevant, because the soft robot is stiff in the longitudinal direction. The actuation redundancy can be easily resolved in an analogous way to Sec. 3.1.2 by introducing the reduced system input $\boldsymbol{F}_{\text{red}} \in \mathbb{R}^{3\times 1}$

where

$$\underbrace{\begin{bmatrix} F_{\mathrm{red},1} \\ F_{\mathrm{red},2} \\ F_{\mathrm{red},3} \end{bmatrix}}_{\boldsymbol{F}_{\mathrm{red}}} = \underbrace{\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}}_{\boldsymbol{\Phi}} \underbrace{\begin{bmatrix} F_{1,\mathrm{up}} \\ F_{1,\mathrm{low}} \\ F_{2,\mathrm{up}} \\ F_{2,\mathrm{low}} \\ F_{3,\mathrm{up}} \\ F_{3,\mathrm{low}} \end{bmatrix}}_{\boldsymbol{F}}. \tag{6.5}$$

The inverse direction of the projection can be defined as

$$\boldsymbol{F} = \boldsymbol{\Phi}^{\dagger} \boldsymbol{F}_{\mathrm{red}} \tag{6.6}$$

with the pseudionverse

$$\boldsymbol{\Phi}^{\dagger} = \begin{bmatrix} \boldsymbol{\Phi}_1^{\dagger} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_2^{\dagger} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_3^{\dagger} \end{bmatrix} \tag{6.7}$$

where

$$\boldsymbol{\Phi}_k^{\dagger} = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & F_{\mathrm{red},k} \geq 0 \\ \begin{bmatrix} 0 \\ -1 \end{bmatrix} & \text{otherwise.} \end{cases} \tag{6.8}$$

$$\tag{6.9}$$

Thus, for each tendon pair only one of the tendons transmits a pulling force, while the other one transmits no force. A positive sign of the reduced tendon force $F_{\mathrm{red},k}$ indicates that the force has to be applied to the upper tendon, a negatives sign indicates that it has to be applied to the lower tendon. The equations of motion then result in

$$\boldsymbol{M}(\boldsymbol{y},t)\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) = \boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{\Phi}^{\dagger}(\boldsymbol{y})\boldsymbol{F}_{\mathrm{red}}(t). \tag{6.10}$$

Note that with the tip position $\boldsymbol{z} \in \mathbb{R}^{2\times 1}$ as the system output and the reduced tendon forces $\boldsymbol{F}_{\mathrm{red}} \in \mathbb{R}^{3\times 1}$ as a system input, the system is still redundantly actuated since the kinematic redundancy remains. In state-space form this can be written as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \tag{6.11}$$

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}). \tag{6.12}$$

Here the state function $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$ is

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{y}} \\ \boldsymbol{M}(\boldsymbol{y},t)^{-1}\left(\boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t) - \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t) + \boldsymbol{B}(\boldsymbol{y})\boldsymbol{\Phi}^{\dagger}\boldsymbol{F}_{\mathrm{red}}\right) \end{bmatrix}. \tag{6.13}$$

The state vector is $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{y} & \dot{\boldsymbol{y}} \end{bmatrix}^\top$, the system input $\boldsymbol{u} = \boldsymbol{F}_{\mathrm{red}}$ are the reduced tendon forces and $\boldsymbol{g}(\boldsymbol{x})$ is the output function. As system output the tip position is chosen.

In the following, it is assumed that all state variables β_i and $\dot{\beta}_i$ are directly accessible for feedback control. In practical applications, often, only the curvatures β_i can be measured directly by sensors integrated into the soft robot, as, e.g., shown in chapter 4 an in [KramerEtAl11, OzelEtAl15, GeEtAl16b, GrubeSeifried22a]. The rate of change $\dot{\beta}_i$ of the curvature usually cannot be measured directly due to the lack of suitable sensors. However, as shown in [GrubeSeifried22a], the rate of change $\dot{\beta}_i$ of the curvature can, e.g., be determined by numerical differentiation with good accuracy.

6.3 Quasi-Static Data-Driven Controller

As a first controller, a quasi-static data-driven feedback control approach for redundantly actuated systems based on [ThuruthelEtAl16b, ThuruthelEtAl17a] is considered. The dynamics of the system are neglected. The controller is based on a neural network representation of the local inverse quasi-static behavior of the soft robot. In the following, first the structure of the controller is described. Then the training data collection for training of the neural network is described.

6.3.1 Controller Structure

The quasi-static forward behavior of the soft robotic test system, which corresponds to the forward kinematics of rigid robots, can be written as

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{u}) \tag{6.14}$$

with the system input $\boldsymbol{u} \in \mathbb{R}^{3\times 1}$ and the tip position $\boldsymbol{z} \in \mathbb{R}^{2\times 1}$ as the system output. For control applications, however, the inverse of the quasi-static behavior is more relevant. Since the soft robot is redundantly actuated, the function $\boldsymbol{h}(\boldsymbol{u})$ is surjective. Thus, there is no unique global solution for the inversion of the forward model (6.14). However, the inverse differential quasi-static behavior

$$\delta\boldsymbol{z} = \boldsymbol{J}(\boldsymbol{u})\delta\boldsymbol{u} \tag{6.15}$$

can be used to obtain a locally unique solution [Siciliano16]. Here, $\boldsymbol{J}(\boldsymbol{u}) \in \mathbb{R}^{2\times 3}$ is the Jacobian of $\boldsymbol{h}(\boldsymbol{u})$ with respect to the system input $\boldsymbol{u}$. The variations of

the tip position $\boldsymbol{z}$ and the system input $\boldsymbol{u}$ are denoted $\delta\boldsymbol{z}$ and $\delta\boldsymbol{u}$, respectively. Following [ThuruthelEtAl17a], equation (6.15) can be discretized such that

$$\boldsymbol{z}_{k+1} - \boldsymbol{z}_k = \boldsymbol{J}(\boldsymbol{u}_k)(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k). \tag{6.16}$$

Here, $\boldsymbol{u}_{k+1}$ is the system input required to reach the tip position $\boldsymbol{z}_{k+1}$ starting from the current tip position $\boldsymbol{z}_k$ with the corresponding system input $\boldsymbol{u}_k$. The index k denotes the k-th timestep. In general, equation (6.16) holds only if the difference between the control variables $\boldsymbol{u}_k$ and $\boldsymbol{u}_{k+1}$ is infinitesimally small. In practice, it is shown in [Siciliano16] for rigid robots that equation (6.16) gives good approximations even for larger distances.

For quasi-static control, equation (6.16) must be solved for the system input $\boldsymbol{u}_{k+1}$. This results in

$$\boldsymbol{u}_{k+1} = \boldsymbol{J}(\boldsymbol{u}_k)^\dagger \left(\boldsymbol{z}_{k+1} - \boldsymbol{z}_k + \boldsymbol{J}(\boldsymbol{u}_k)\boldsymbol{u}_k\right) = \boldsymbol{g}(\boldsymbol{z}_k, \boldsymbol{z}_{k+1}, \boldsymbol{u}_k) \tag{6.17}$$

where $\boldsymbol{J}(\boldsymbol{u}_k)^\dagger$ is the Moore-Penrose pseudoinverse of the Jacobian $\boldsymbol{J}$. In the following, the mapping function from $\boldsymbol{z}_k$, $\boldsymbol{z}_{k+1}$ and $\boldsymbol{u}_k$ to $\boldsymbol{u}_{k+1}$ is denoted as $\boldsymbol{g}$. It represents a locally valid inversion of the quasi-static behavior of the soft robot. For the soft robotic test system considered here, the function of the local inverse quasi-static behavior $\boldsymbol{g}$ cannot be determined analytically. Therefore, it is approximated by a neural network, which is visualized in Fig. 6.7. The inputs of the neural network are the current tip position $\boldsymbol{z}_k \in \mathbb{R}^{2\times 1}$, the current system input $\boldsymbol{u}_k \in \mathbb{R}^{3\times 1}$, and the desired tip position for the next timestep $\boldsymbol{z}_{\mathrm{ref},k+1} \in \mathbb{R}^{2\times 1}$ where $\boldsymbol{z}_{\mathrm{ref},k+1} = \boldsymbol{z}_{\mathrm{ref}}(t_{k+1})$. The output of the neural network is the system input $\boldsymbol{u}_{k+1} \in \mathbb{R}^{3\times 1}$ needed to reach the desired tip position. The control error is not directly fed into the neural network, but for slow motions the neural network can derive it from the current tip position $\boldsymbol{z}_j$ and the desired tip position for the next timestep $\boldsymbol{z}_{j+1}$. Since the desired tip position is fed directly into the controller, the controller is not a pure feedback controller, but a combination of a feedforward and a feedback controller. The influence of the feedforward and feedback parts of the controller is set indirectly by the choice of training data and training parameters of the neural network and also depends on the system. The neural network has an input layer with seven neurons, a hidden layer with 30 neurons and tanh activation functions, and an output layer with three neurons and linear activation functions. The neural network is implemented in MATLAB using the "Deep Learning Toolbox". During training of the neural network Bayesian regularization is used.

This results in the controller structure shown in Fig. 6.8. The controller runs with a sample rate of $f_{\mathrm{controller}} = 5\,\mathrm{Hz}$. The reason for this relatively low frequency is the quasi-static behavior of the controller, which neglects the dynamics of the soft robot. Every change of the control output $\boldsymbol{u}$ excites the dynamics of the system, but the quasi-static controller assumes that the steady-state position is reached instantaneously. With the low sampling frequency, the transients can at

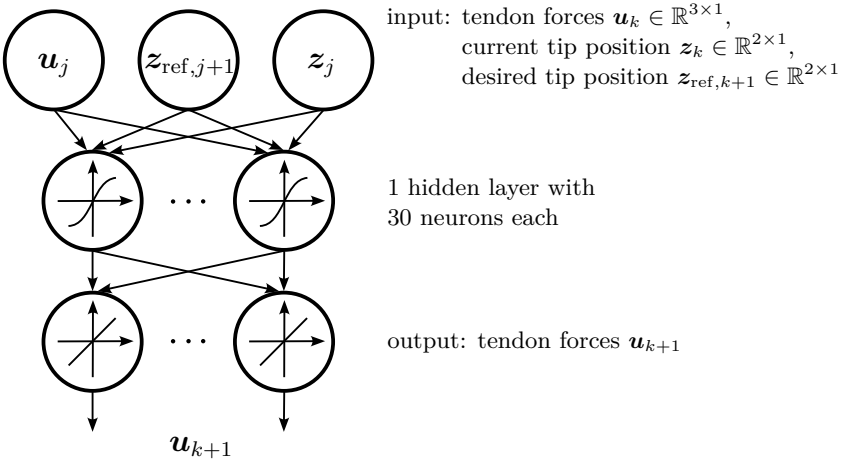


Figure 6.7: Structure of the neural network of the quasi-static controller.

least partially decay before the measurements for the next control step are taken, which improves the performance and stability of the controller. Conversely, if the sample rate of the controller is chosen even lower, only very slow movements of the robot are possible.

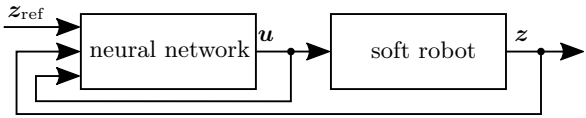


Figure 6.8: Controller structure of the quasi-static neural network based controller.

6.3.2 Training Data Collection

In order to train the neural network, a data set with a sufficient number of value pairs of the mapping $(\boldsymbol{z}_k, \boldsymbol{z}_{\mathrm{ref},k+1}, \boldsymbol{u}_k) \rightarrow \boldsymbol{u}_{k+1}$ is required. The simulation model derived in Sec. 6.2 is used to collect the training data. First, to generate a pair of values, the system input $\boldsymbol{u}_k$ and the varied system input $\boldsymbol{u}_{k+1}$ are randomly chosen. The values for the system input are chosen such that a maximum tendon force of $N_{\max} = 20\,\mathrm{N}$ is not exceeded and the change in the tendon force is not too large, which is enforced by

$$||\boldsymbol{u}_k - \boldsymbol{u}_{k+1}|| < \varepsilon N_{\max} \tag{6.18}$$

with $\varepsilon = 5\,\%$.

Second, the initial system input $\boldsymbol{u}_k$ is applied to the simulation model of the soft robotic test system and the resulting initial position $\boldsymbol{z}_k$ is determined after the transient processes have decayed. Third, the system input is changed to the new system input $\boldsymbol{u}_{k+1}$, resulting in the corresponding position $\boldsymbol{z}_{\mathrm{ref},k+1}$. In this way a complete value pair of the mapping $(\boldsymbol{z}_k, \boldsymbol{z}_{\mathrm{ref},k+1}, \boldsymbol{u}_k) \rightarrow \boldsymbol{u}_{k+1}$ is determined. This procedure is repeated 10,000 times to collect the training data for the neural network. The entire data collection process takes about 5 min on a PC with an "Intel Core i7 - 6700K" processor. The resulting workspace of the soft robot is shown in Fig. 6.9.

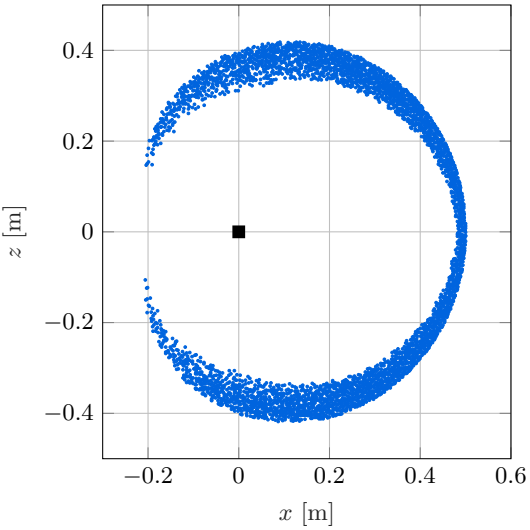


Figure 6.9: Empirically determined workspace of the quasi-static controller for a maximum tendon force of $N_{\max} = 20\,\mathrm{N}$.

6.4 Linear Quadratic Regulator with Integral Action and Gain Scheduling

As a second controller a linear quadratic controller with integral action (LQI controller) and gain scheduling is considered. In general, for the control of linear time-invariant systems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \tag{6.19}$$

$$\boldsymbol{z} = \boldsymbol{C}\boldsymbol{x} \tag{6.20}$$

LQI controllers can be used [YoungWillems72]. Here, $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ the system input and $\boldsymbol{z}$ the system output. The dynamic behavior of the linear system and the relationship between the input vector $\boldsymbol{u}$, the states $\boldsymbol{x}$ and the output $\boldsymbol{z}$ are described by the system matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$. LQI controllers are not further described here because they are well established and extensively discussed in the literature, see e.g. [Porter71, HendricksEtAl08].

LQI controllers require a linear model of the system to be controlled for the design. However, soft robots, including the soft robotic test system considered here, typically exhibit nonlinear behavior, which can be described in state space as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{6.21}$$

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}). \tag{6.22}$$

In order to derive an LQI controller for such a nonlinear system, it has to be linearized. If the influence of the nonlinearities on the system behavior is sufficiently small, the system can simply be linearized around a stationary point. However, for the soft robotic test system, as for many soft robots, the influence of the nonlinearites is too large for this. A relatively simple and established method of designing controllers for such systems is gain scheduling [Adamy14]. The system is linearized in several operating points and a linear controller is designed for each of these operating points OP. Each operating point is defined by its state vector $\boldsymbol{x}_{\mathrm{OP}}$, the system input $\boldsymbol{u}_{\mathrm{OP}}$, and the corresponding system output $\boldsymbol{z}_{\mathrm{OP}}$. During a control process, interpolation is then performed between these controllers. In the following, trilinear interpolation is used. The linearized system at the operating point OP results in

$$\widetilde{\dot{\boldsymbol{y}}} = \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{y}} + \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{u}}, \tag{6.23}$$

$$\widetilde{\boldsymbol{z}} = \widetilde{\boldsymbol{C}}\widetilde{\boldsymbol{y}} \tag{6.24}$$

with $\widetilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{y}_{\mathrm{OP}}$, $\widetilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}_{\mathrm{OP}}$ and $\widetilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{OP}}$. The matrices of the state

space representation of the linearized system are

$$\widetilde{\boldsymbol{A}} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}_{\mathrm{OP}},\boldsymbol{u}_{\mathrm{OP}}}, \tag{6.25}$$

$$\widetilde{\boldsymbol{B}} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{y}_{\mathrm{OP}},\boldsymbol{u}_{\mathrm{OP}}}, \tag{6.26}$$

$$\widetilde{\boldsymbol{C}} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}_{\mathrm{OP}},\boldsymbol{u}_{\mathrm{OP}}}. \tag{6.27}$$

An LQI controller can now be derived for each of the linearized systems in the operating points OP.

To apply the gain scheduling approach to the soft robotic test system, the nonlinear system (6.11) and (6.12) is linearized around 2288 operating points OP that are regularly distributed in the workspace. The operating points are chosen so that the velocities of the soft robot at these points are zero. In general, each operating point is defined by its state vector $\boldsymbol{x}_{\mathrm{OP}}$, the system input $\boldsymbol{u}_{\mathrm{OP}}$, and the corresponding system output $\boldsymbol{z}_{\mathrm{OP}}$. For the soft robotic test system considered here, the number of parameters needed to describe an operating point can be reduced to three if the influence of external forces acting on the system is assumed to be small. In this case, the steady-state configuration of the soft robot depends only on the three tendon forces $F_{\mathrm{red},1} \dots F_{\mathrm{red},3}$. In the following, the curvatures of the first, third, and fifth segments, β_1, β_3 and β_5, are chosen as scheduling parameters. These are collected in the scheduling vector $\boldsymbol{\sigma}(\boldsymbol{x}_{\mathrm{OP}}) = [\beta_1\, \beta_3\, \beta_5]^\top$. The operating points used here are shown in Fig. 6.10. Note that because the operating points are defined in a different way than the workspace for the quasi-static controller shown in Fig. 6.9, the space spanned by the operating points is slightly different.

To control the soft robot, at each time step a trilinear interpolation is performed between the eight controllers spanning the cuboid in the three-dimensional parameter space where the current scheduling parameter $\boldsymbol{\sigma}$ lies. The structure of the LQI controller with gain scheduling is shown in Fig. 6.11. The control law results in

$$\boldsymbol{u}(\boldsymbol{\sigma}) = \boldsymbol{K}\,(\boldsymbol{\sigma})\,(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{OP}}) + \boldsymbol{K}_{\mathrm{i}}(\boldsymbol{\sigma}) \int (\boldsymbol{z} - \boldsymbol{z}_{\mathrm{ref}})\mathrm{d}t + \boldsymbol{u}_{\mathrm{OP}}(\boldsymbol{\sigma}). \tag{6.28}$$

The gain matrix of the controller is composed of the gain $\boldsymbol{K}_i$ of the integrated control error $\boldsymbol{x}_e = \int(\boldsymbol{z} - \boldsymbol{z}_{\mathrm{ref}})$ for the integral behavior and the gain matrix $\boldsymbol{K}$ for the proportional and derivative behavior of the controller.

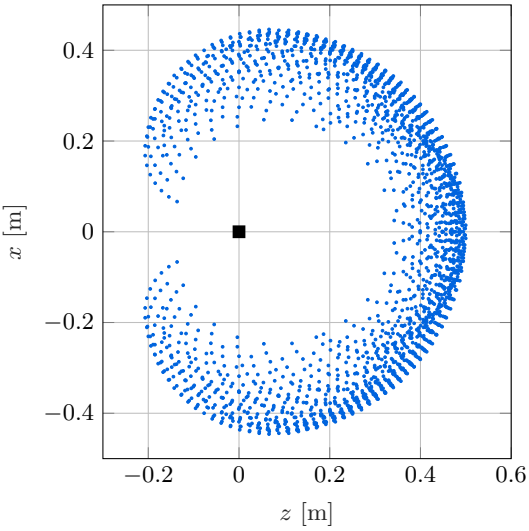


Figure 6.10: Tip position $\boldsymbol{z}$ of the operating points used for gain scheduling.

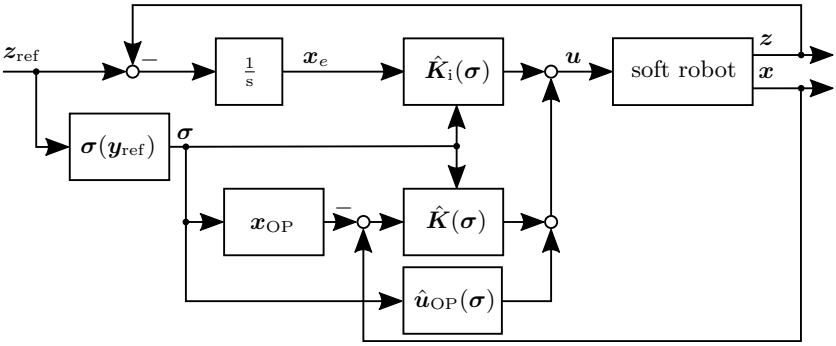


Figure 6.11: Controller structure of the LQI controller with gain scheduling.

6.5 Model Predictive Controller

As a third control method a model predictive controller (MPC) [RichaletEtAl78] is considered. The controller structure is shown in Fig. 6.12. Model predictive controllers rely on an accurate dynamic model of the system that is used to predict the system behavior for the next few timesteps. At each time step k, an optimization problem is solved to find the best system input for the next time

step $k+1$ with respect to some cost function J. The MPC optimizes the system input over the control horizon, which has a length of c timesteps, so that the cost function is minimized over the prediction horizon, which has a length of p timesteps. A common choice for the cost function is the weighted sum of the control error $\boldsymbol{e}$ and the control effort. The control effort consists of the magnitude of the system input $\boldsymbol{u}$ and the magnitude of the change $\Delta\boldsymbol{u}$ in the system input. The optimization problem can be formulated as

$$\min_{\boldsymbol{p}\in\mathcal{P}} J(\boldsymbol{x}(k),\boldsymbol{u}(\boldsymbol{p})) \quad \mathcal{P}=\{\boldsymbol{p}\in\mathbb{R}^{m\cdot c}\} \tag{6.29}$$

with the cost function

$$\begin{aligned} J(\boldsymbol{x}(k),\boldsymbol{u}) = \sum_{i=1}^{p} \Big(&\boldsymbol{e}(k+i)^\top \boldsymbol{Q}\boldsymbol{e}(k+i) + \boldsymbol{u}(k+i)^\top \boldsymbol{R}\boldsymbol{u}(k+i) \\ &+ \Delta\boldsymbol{u}(k+i)^\top \boldsymbol{R}_\Delta \Delta\boldsymbol{u}(k+i)\Big) \end{aligned} \tag{6.30}$$

where

$$\boldsymbol{e}(n) = \boldsymbol{z}_{\mathrm{ref}}(n) - \boldsymbol{z}(n), \tag{6.31}$$

$$\Delta\boldsymbol{u}(n) = \boldsymbol{u}(n) - \boldsymbol{u}(n-1). \tag{6.32}$$

Here, $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{R}_\Delta$ are weighting matrices for the different cost terms, respectively. The variables to be computed by optimization are the system inputs $\boldsymbol{u}_{k+1},\ldots,\boldsymbol{u}_{k+c}$ at the next c timesteps. These are collected in the vector

$$\boldsymbol{p} = \begin{bmatrix}\boldsymbol{u}_{k+1}^\top & \boldsymbol{u}_{k+2}^\top & \cdots & \boldsymbol{u}_{k+c}^\top\end{bmatrix}^\top. \tag{6.33}$$

Here, $\boldsymbol{u}_{k+n}$ is the system input that applies to the time step with index n in the control horizon. In total, $m\cdot c$ variables have to be computed in each iteration of the optimization, where m is the number of system inputs and c is the length of the control horizon. To solve the optimization problem, in every timestep, the nonlinear soft robot model (6.11) and (6.12) integrated into the controller has to be evaluated several times. After solving the optimization problem, the system input $\boldsymbol{u}_{k+1}$ is applied to the system for timestep $k+1$.

In the following, a sample time of $T_\mathrm{s}=0.1\,\mathrm{s}$, a prediction horizon of $p=5$ steps and a control horizon of $c=3$ steps are selected. Typically, short sample times and long prediction and control horizons lead to better control results. However, the optimization problem is harder to solve at the cost of higher computational load. The MPC is implemented in Simulink using the "Model Predictive Control" Toolbox. The internal simulation model (6.11) and (6.12) is solved using the explicit Euler method with stepsize T_s. The Matlab function "fmincon" is used to solve the optimization problem. It uses the SQP method for minimization [NocedalWright06]. The MPC is implemented so that the trajectory can be set online, e.g. with joystick commands. The disadvantage is that it is not possible to include future values of the trajectory, which is possible with other implementations of the MPC. Therefore, the chosen implementation of the MPC in Simulink induces a time delay of one time step (0.1 s).

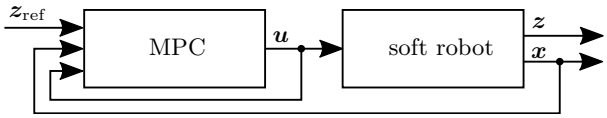


Figure 6.12: Block diagram of the closed control loop with the model predictive controller.

6.6 Evaluation of the Controller Performance

In the following, the tip position trajectory tracking control performance of all three controllers is examined. Here, the control error of the tip position is calculated as

$$e(t) = ||\boldsymbol{z}(t) - \boldsymbol{z}_{\text{ref}}(t)|| \,. \tag{6.34}$$

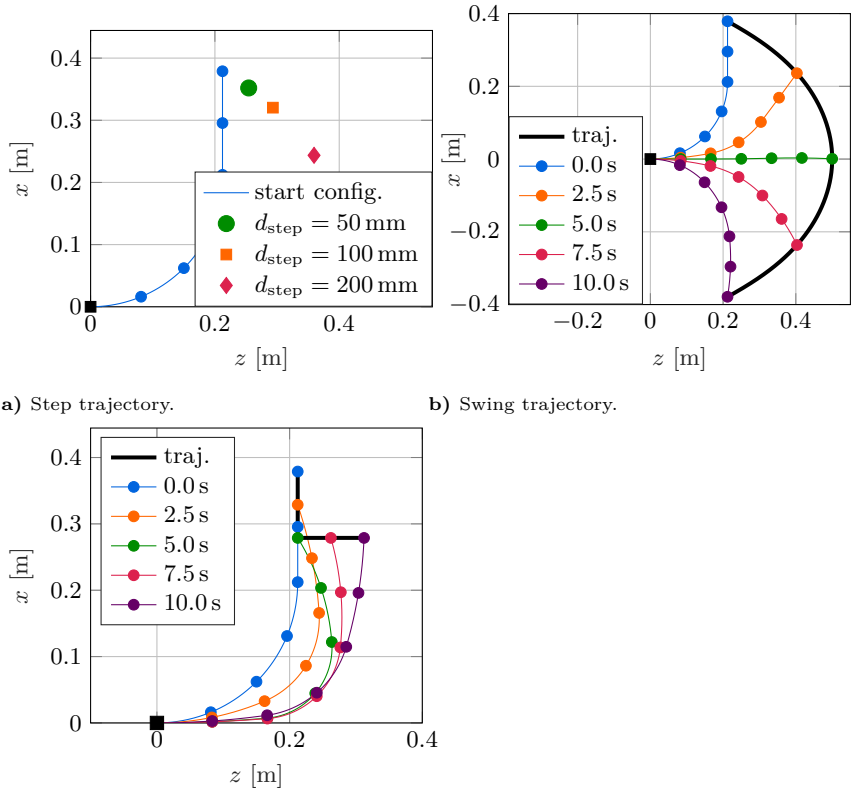
In the following, first the test trajectories used for the evaluation of the performance of the three controllers are presented. Then the trajectory tracking performance of the three controllers is evaluated and compared. Finally, the robustness of the three controllers against parameter uncertainties of the simulation model is examined.

6.6.1 Test Trajectories

To evaluate the performance of the controllers, a step trajectory, a swing trajectory through almost the entire workspace and an L-shaped trajectory are considered. The trajectories are shown in Fig. 6.13. For all trajectories only the tip position is specified, the controllers are free to choose any configuration of the soft robot that reaches that tip position. The configurations shown in Fig. 6.13 are obtained with the LQI controller.

The step trajectory is the simplest of the three trajectories examined and is a standard test trajectory widely used in both linear and nonlinear control. Three different step widths of $d_{\text{step}} = 50\,\text{mm}$, $d_{\text{step}} = 100\,\text{mm}$ and $d_{\text{step}} = 200\,\text{mm}$ are considered. The step trajectory is not continuous, so the trajectory cannot be followed exactly. The swing trajectory is the most natural trajectory for the soft robotic test system considered here. It starts at rest at the start point, traverses almost the entire workspace in a swing movement at a constant velocity of $0.1\,\text{m/s}$ within $10\,\text{s}$, and comes to rest at the end point. The trajectory is not differentiable at its start and end points because there is a jump in the velocity profile. Therefore, the trajectory cannot be followed exactly. The L-shaped trajectory is the most challenging of the three trajectories considered as it is less natural for the soft robotic test system and can only be achieved with large curvatures. It starts from a rest position, followed by two axis-parallel movements at constant velocity, each taking $5\,\text{s}$, and comes to rest again in the final position.

This trajectory is not differentiable at the start and end points and at the edge in the middle of the trajectory. Therefore, it cannot be followed exactly.



a) Step trajectory.

b) Swing trajectory.

c) L-shape trajectory.

Figure 6.13: Examined trajectories.

6.6.2 Trajectory Tracking Control Performance of the Controllers

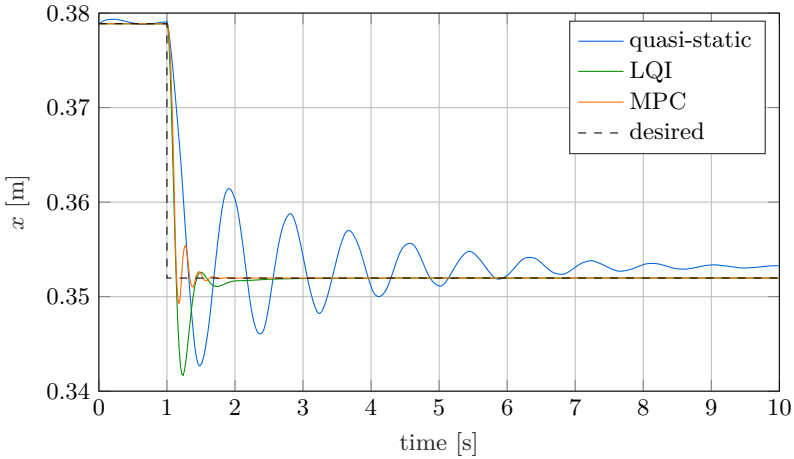
In the following, the trajectory tracking control performance of the three controllers applied to the three trajectories is discussed.

Step Trajectory

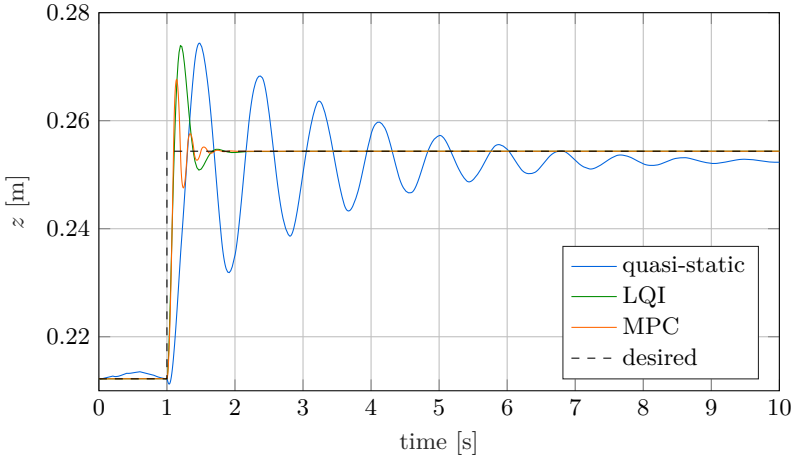
As a first scenario the step trajectory with the three different step widths $d_{\text{step}} = 50\,\text{mm}$, $d_{\text{step}} = 100\,\text{mm}$ and $d_{\text{step}} = 200\,\text{mm}$ is considered. In Fig. 6.14 the x and y coordinates of the tip position are plotted over time for the controllers examined and for a step width of $d_{\text{step}} = 50\,\text{mm}$. The plots for a step width of $d_{\text{step}} = 100\,\text{mm}$ and $d_{\text{step}} = 200\,\text{mm}$ are qualitatively comparable and therefore omitted here. It can be seen that all three controllers are able to successfully move the tip of the soft robot to the desired reference position. The overall performance of the MPC and LQI controller is similar, while the quasi-static controller shows a much worse performance. Both the MPC and LQI controller show a negligible steady-state error. For the LQI, this can be explained by its integral behavior. The MPC achieves a vanishing steady state error because its internal prediction model exactly matches the simulation model. The quasi-static controller shows a steady state error of $2\,\% \ldots 5\,\%$. This can be explained by the fact that the neural network approximation of the quasi-static behavior used by the controller is not exact.

The step response of the three controllers can be described by the rise time, settling time, overshoot and remaining error. These quantities are listed in Tabs. 6.2 to 6.4 for the three different step widths and the three different controllers. The rise time is defined as the time taken for the response to rise from $10\,\%$ to $90\,\%$ of the difference between the initial value and the steady state value. The settling time is defined as the time taken for the system to settle within a band of $2\,\%$ of the difference between the initial value and the steady state value around the final rest position. For all step sizes considered, the rise time of the quasi-static controller is more than twice as long as for the two dynamic controllers, and the settling time is about an order of magnitude longer. This can be explained by the fact that the quasi-static controller neglects the dynamics. Therefore, it cannot actively damp the oscillations excited by the step trajectory. The rise time of the MPC and LQI controller is similar. However, the settling time of the MPC is $15\,\% \ldots 45\,\%$ lower than that of the LQI controller, depending on the step width. The overshoot of the MPC is also $30\,\% \ldots 60\,\%$ lower than for the other two controllers. Both can be explained by the predictive behavior of the MPC.

The behavior of the controllers is similar for the three different step widths. Only the settling time increases clearly with increasing step size and is $2 \ldots 3$ times greater for a step size of $d_{\text{step}} = 200\,\text{mm}$ than for a step size of $d_{\text{step}} = 50\,\text{mm}$.



a) x-coordinate.



b) y-coordinate.

Figure 6.14: Reference trajectory as well as the tip position for different controllers with $d_{\text{step}} = 50\,\text{mm}$ over time.

Table 6.2: Characteristic quantities of the step trajectory with $d_{\text{step}} = 50\,\text{mm}$.

	Quasi-Static	**LQI**	**MPC**
rise time (10% . . . 90%)	237.3 ms	90.2 ms	86.2 ms
settling time (2% band)	7.65 s	0.88 s	0.49 s
overshoot	47.46%	46.36%	31.49%
steady-state error	4.85%	<0.01%	<0.01%

Table 6.3: Characteristic quantities of the step trajectory with $d_{\text{step}} = 100\,\text{mm}$.

	Quasi-Static	**LQI**	**MPC**
rise time (10% . . . 90%)	219.2 ms	86.0 ms	85.9 ms
settling time (2% band)	9.81 s	1.83 s	1.49 s
overshoot	50.21%	45.41%	27.24%
steady-state error	5.12%	<0.01%	<0.01%

Table 6.4: Characteristic quantities of the step trajectory with $d_{\text{step}} = 200\,\text{mm}$.

	Quasi-Static	**LQI**	**MPC**
rise time (10% . . . 90%)	248.9 ms	78.6 ms	66.6 ms
settling time (2% band)	14.10 s	1.61 s	1.37 s
overshoot	53.71%	54.84%	20.51%
steady-state error	2.10%	<0.01%	<0.01%

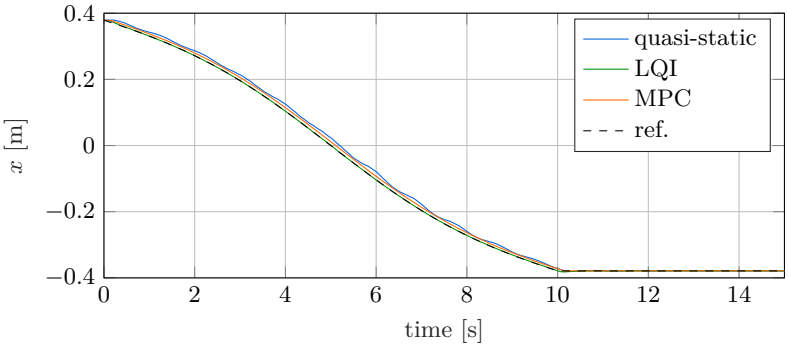
Swing Trajectory

As a second simulation scenario the swing trajectory with a constant velocity of 0.1 m/s is considered. In Fig. 6.15 the x- and y-coordinate of the tip position and the control error e are plotted over time for the three examined controllers. It can be seen that the three controllers are able to follow the trajectory. However, there are clear differences in the control error.

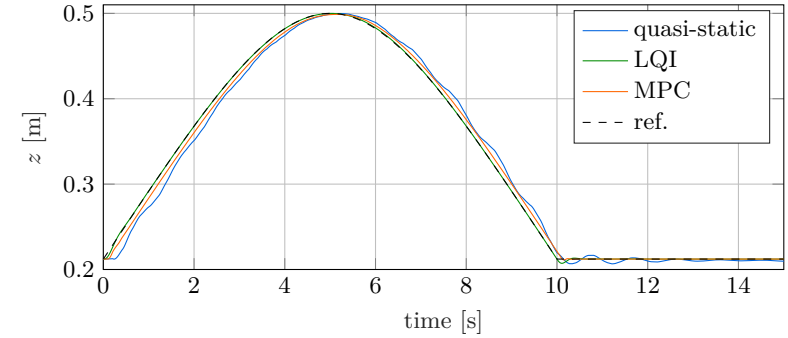
The quasi-static controller has the largest control error of the three controllers with a maximum error of 27.3 mm (5.46 % related to the length of the soft robot). This is more than twice the maximum error of the MPC and four times the maximum error of the LQI controller. The control error of the quasi-static controller consists of three parts: First, there is an offset during the movement between the desired and the actual trajectory, which results from the low sample rate of the controller. Second, it can be seen from the plots that the quasi-static controller oscillates around the trajectory, which can be explained by the controller neglecting the dynamics of the system. And third, the neural network approximation of the quasi-static behavior, which is not perfect, adds to the error. Note that while for physical systems, data-driven methods usually have the advantage over model-based methods of being able to include hard-to-model effects, this is not the case here because only simulations are performed. Here, the mechanical models on which the LQI controller and the MPC are based are exactly the same as the simulation model. However, the data-driven model learned by the quasi-static controller is only an approximation of the simulation model. The quasi-static controller also takes the longest time to reach the end position after the end of the swing motion. Within the additional time of 5 s considered after the end of the motion, it does not succeed in reaching the target position completely. If a longer simulation time is considered, the oscillation decays due to the internal damping of the soft robot, and a permanent error of 1.7 mm (0.34 %) remains.

The LQI controller has the smallest control error of the three controllers examined. The largest control errors occur at the beginning ($t = 0$ s) and at the end ($t = 10$ s) of the swing movement with a maximum control error of 6.13 mm (1.23 %). This can be explained by the fact that the trajectory is not differentiable at these points, as described in Sec. 6.6.1. For the rest of the motion, the control error is less than 1 mm (0.2 %).

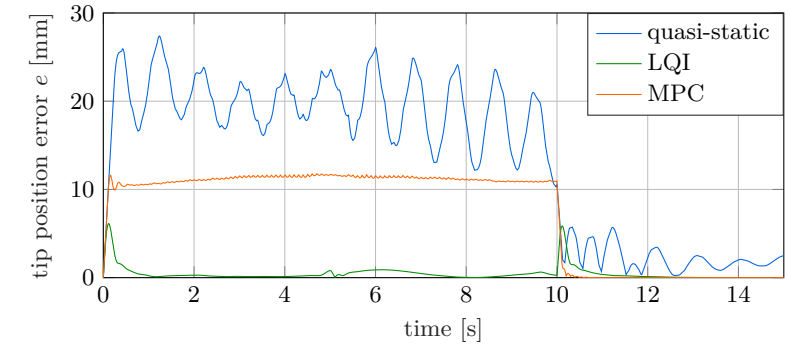
The MPC has a maximum control error of 11.77 mm (2.35 %). During the swing movement the error is nearly constant with a variation of only about 1 mm. This can be explained by a delay between the desired and the actual trajectory of one time step, which is caused by the chosen implementation of the MPC. This is discussed in more detail in Sec. 6.5. At the end of the swing motion, when the reference signal is constant, the control error disappears almost completely as the influence of the induced time delay vanishes.



a) x-coordinate.



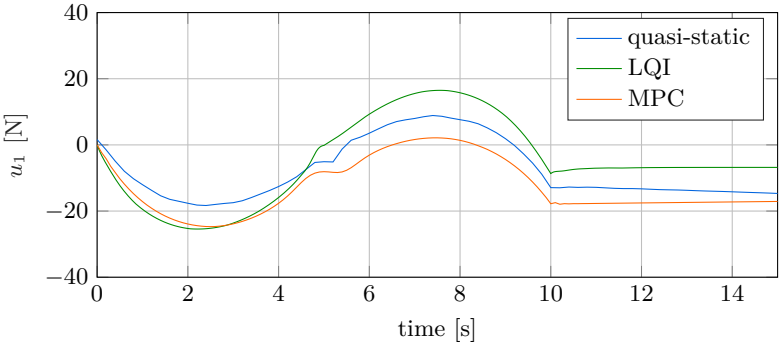
b) z-coordinate.



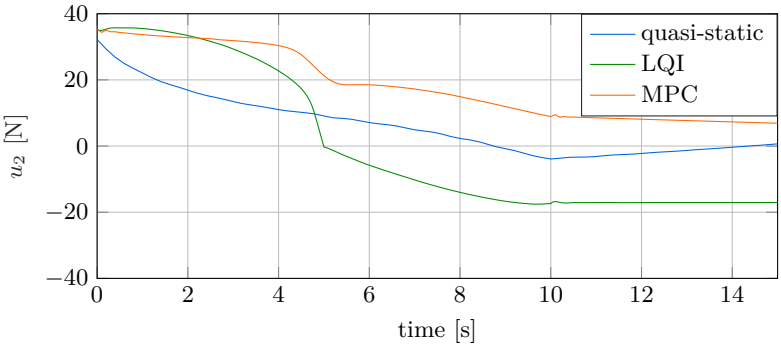
c) Error.

Figure 6.15: Tip trajectory, tip position and tip position control error for different controllers for the swing trajectory over time.

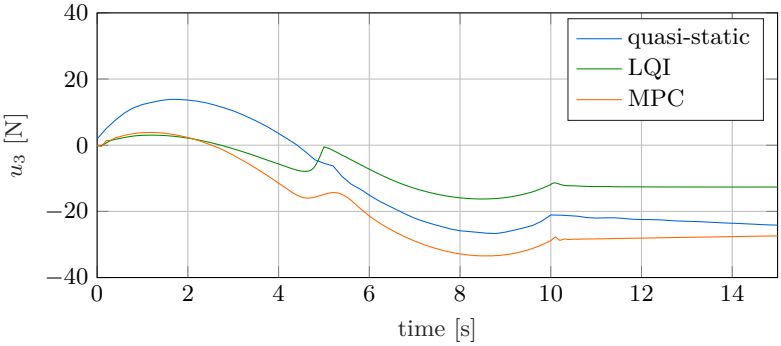
In addition to the control error, the system input applied by the different controllers is also of interest. The actuation forces applied to the three tendon pairs by the examined controllers are shown in Fig. 6.16. Note that a positive force corresponds to actuation of the upper tendon and a negative force corresponds to actuation of the lower tendon, as described in Sec. 6.2.2. It can be seen that the overall course of the tendon forces is similar for all three controllers. However, there are notable differences with differences of up to 29 N between the tendon forces calculated by the three controllers investigated. The different actuation forces can be explained by the redundancy of the test system, which allows the trajectory to be followed with different configurations of the robot. This can also be seen in Fig. 6.17, which shows the configuration of the robot for the end position at time $t = 15\,\mathrm{s}$. Here the configuration obtained with the LQI controller shows a stronger curvature at the beginning of the soft robot arm and a smaller curvature at its tip, while the MPC shows the opposite. The configuration of the quasi-static controller is in between, but closer to the configuration obtained by the MPC.



a) Tendon force u_1 of tendon pair 1.



b) Tendon force u_2 of tendon pair 2.



c) Tendon force u_3 of tendon pair 3.

Figure 6.16: Actuation forces for the swing trajectory over time.

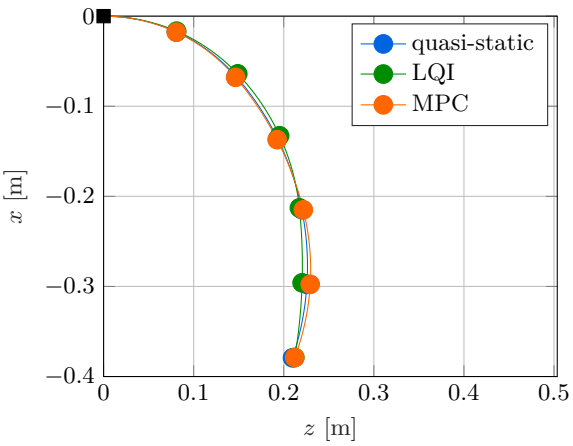


Figure 6.17: Configurations of the soft robot in the end position of the sweep trajectory for the three controllers.

L-Shape Trajectory

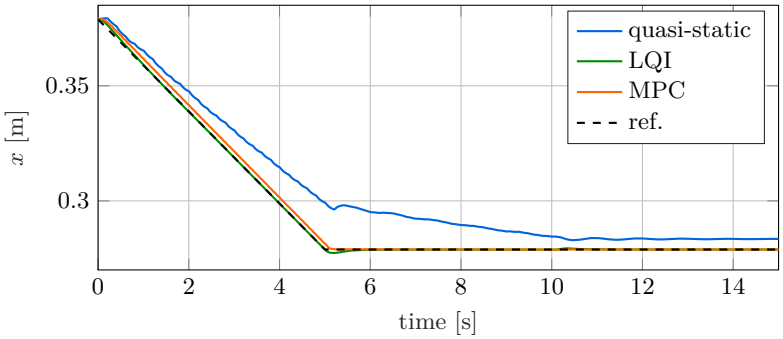
The results for the L-shape trajectory are shown in Figs. 6.18 and 6.19. The tip trajectories, the trajectory tracking control error and the control inputs are presented for the three different controllers. The overall results are similar to those for the swing trajectory. However, for the dynamic controllers an even lower control error could be obtained.

As for the swing trajectory, the control error for the quasi-static controller is much larger than for the two dynamic controllers. The maximum control error of 21.57 mm (4.31 % related to the length of the soft robot) is at the same level as for the swing trajectory. The control error is largest in the middle of the trajectory. This can be explained by the fact that here large curvatures of the soft robot are required to follow the trajectory. These require high tendon forces, which are partly not represented in the training set of the controller and therefore force the neural network to extrapolate which leads to a bad controller performance. If the time simulation is continued for a longer simulation time, a permanent control error of 5.47 mm (1.09 %) is achieved, which is much higher than for the swing trajectory.

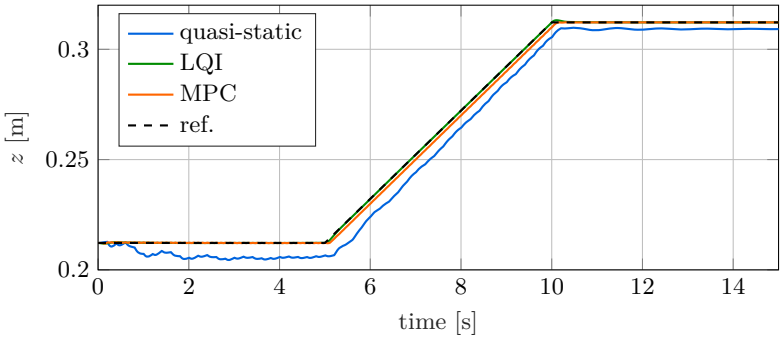
For the LQI controller a maximum control error of 1.71 mm (0.34 %) could be achieved, which is about one third of the control error achieved for the swing trajectory. As for the swing trajectory, the control error is largest where the trajectory is not differentiable. The lower control error compared to the swing trajectory can mainly be explained by the fact that the L-trajectory has a lower velocity, which also leads to smaller jumps in the velocity profile of the trajectory.

Except for the three sudden spikes and the subsequent decay at the points where the trajectory is not differentiable, the control error of the LQI controller is less than 0.2 mm (0.4 %).

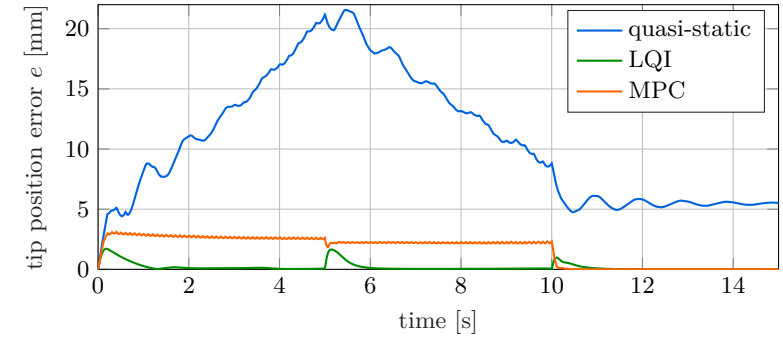
As for the swing trajectory, the control error of the MPC is mainly due to the lag, resulting in a nearly constant control error over time. Since the velocity of the L-shaped trajectory is lower than that of the swing trajectory, a lower maximum control error of only 3.11 mm (0.62 %) could be obtained. Again, this is supported by the fact that the control error quickly approaches zero as soon as the reference trajectory has reached the end position.



a) x-coordinate.

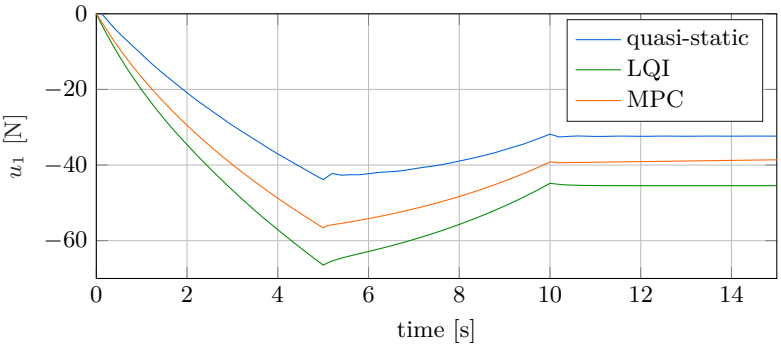


b) z-coordinate.

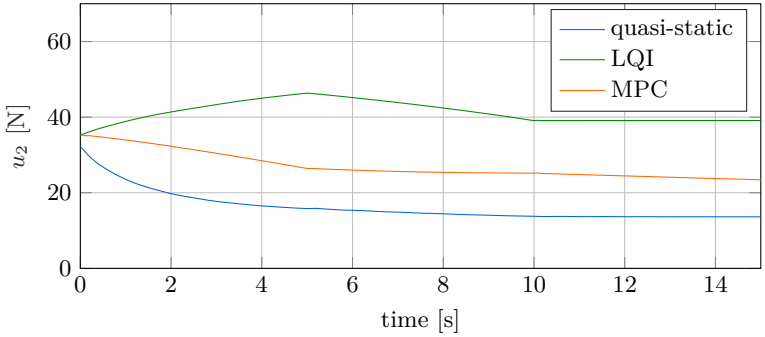


c) Error.

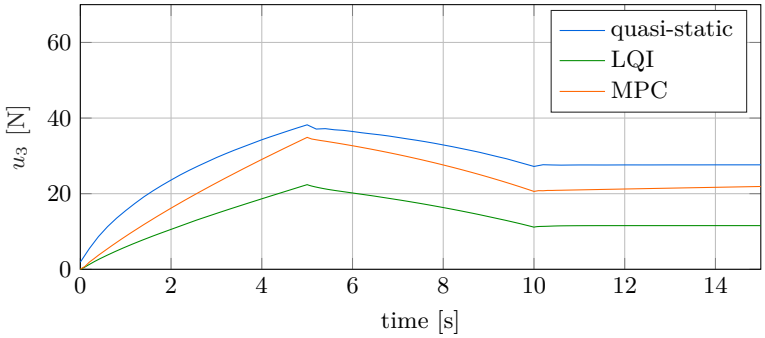
Figure 6.18: Tip trajectory, tip position and tip position control error for different controllers for the L-shape trajectory over time.



a) Tendon force u_1 of tendon pair 1.



b) Tendon force u_2 of tendon pair 2.



c) Tendon force u_3 of tendon pair 3.

Figure 6.19: Actuation forces for the L-shape trajectory over time.

6.6.3 Parameter uncertainty

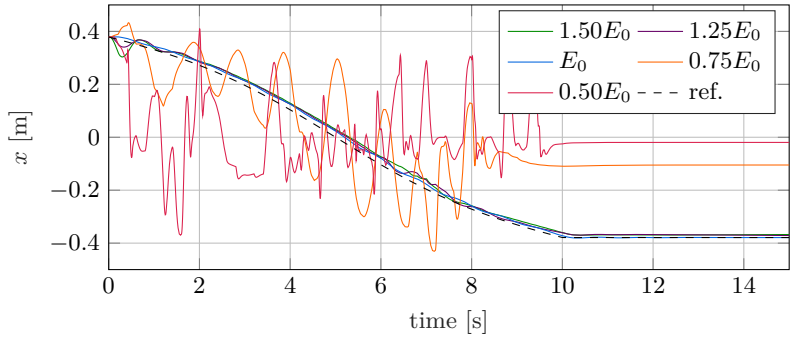
Finally, the robustness of the different controllers to variations in the material parameters of the robot is investigated. In soft robotics, the material parameters are often not well known, contain unmodeled nonlinear effects, are difficult to determine, and may even change over time [KhanEtAl20]. Therefore, robustness of the controllers against parameter uncertainty is important.

In the following, as an example variations of the Young's modulus E are considered. The Young's modulus usually has a large influence on the quasi-static and the dynamic behavior of soft robots and at the same time is often hard to determine for soft material. In a first step, the three controllers are designed using a model with a nominal Young's modulus $E_0 = 7.32 \cdot 10^5\,\mathrm{Pa}$. Next, five simulations are performed for each of the three controllers with different values of the Young's modulus respectively. The nominal value of Young's modulus is considered as well as a variation of $\pm 25\,\%$ and $\pm 50\,\%$. Note that a change in Young's modulus also affects the damping, since stiffness-proportional damping is assumed here. In the following, only the trajectory tracking of the swing trajectory is considered in more detail as an example. For the other trajectories, qualitatively comparable results could be obtained. In Figs. 6.20 to 6.22 the x- and y-coordinate of the tip position is plotted over time for the three controllers and different values of the Young's modulus E of the simulation model. These are discussed in the following.

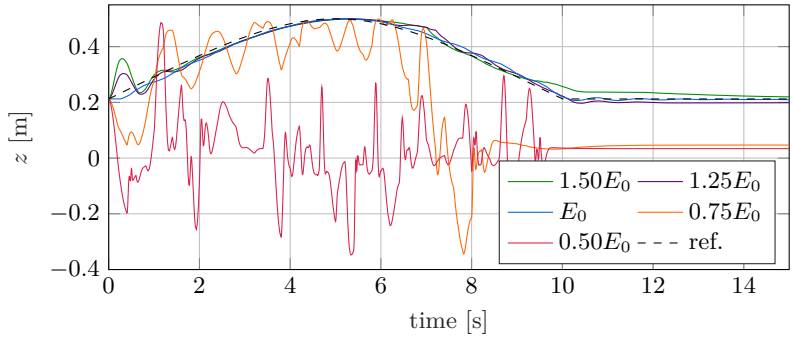
Quasi-Static Controller

The trajectory tracking control results for the quasi-static controller are shown in Fig. 6.20. The x- and y-coordinate of the tip position as well as the tip position control error e are plotted over time for the different values of the Young's modulus E. The control error is shown only for those values of Young's modulus for which successful trajectory tracking was possible. The quasi-static controller shows the worst results of the three controllers considered. For an increased Young's modulus of the simulation model, the quasi-static controller shows a similar behavior as for the nominal Young's modulus. There are minor deviations only at the beginning of the trajectory, which can be explained by the controller needing some time to compensate for the changed Young's modulus. However, for a reduced value of Young's modulus in the simulation model, an unsteady oscillatory behavior of the tip position is obtained during the swing motion. After the reference position stops changing, the tip position comes to rest but shows a very large remaining control error. The oscillatory behavior can be explained by the fact that due to the lower Young's modulus, the tendon forces applied by the controller lead to a much larger deflection of the soft robot than expected by the controller. Since the controller is too slow to compensate for this, the tip clearly leaves the workspace considered during training of the neural network on which the quasi-static controller is based. In addition, these large movements

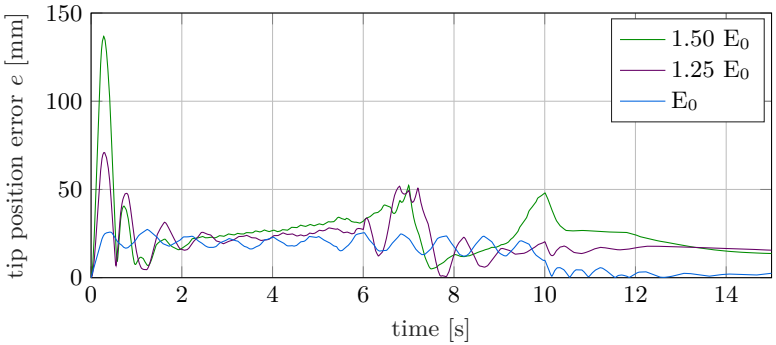
strongly excite the dynamics of the system, which the quasi-static controller cannot compensate for. This is further intensified by the fact that, in addition to stiffness, damping is also reduced, since stiffness-proportional damping is assumed. As a result, the controller cannot determine the required system input sufficiently accurately, which leads to the unsteady oscillatory behavior. On the other hand, if the Young's modulus of the simulation model is larger than that assumed by the controller, the initial system input is too small. The tip of the robot does not reach the desired position but remains in the workspace and the dynamics are only slightly excited. This allows the controller to subsequently compensate for the control error, resulting in a controller performance that is almost as good as for the system with the nominal Young's modulus. Again, remember that stiffness proportional damping of the soft robot is assumed. Therefore, with an increase in Young's modulus, the value for the internal damping of the soft robot also increases, which supports controller performance.



a) x-coordinate.



b) z-coordinate.

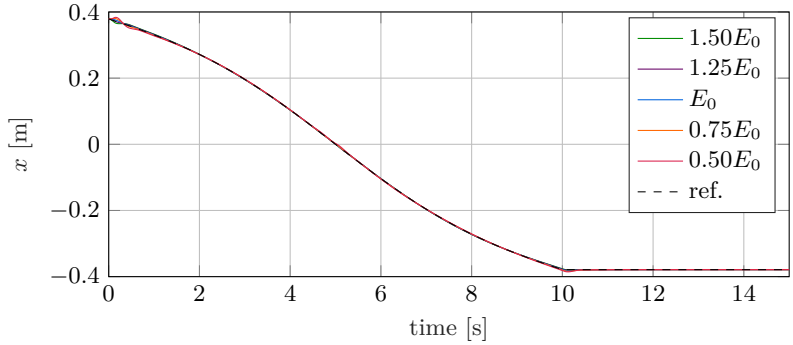


c) Error.

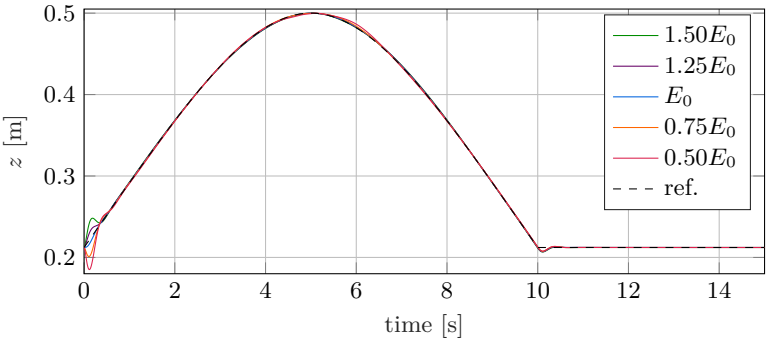
Figure 6.20: Trajectories of the tip position and control error for the quasi-static controller for different values of the Young's modulus E.

LQI Controller

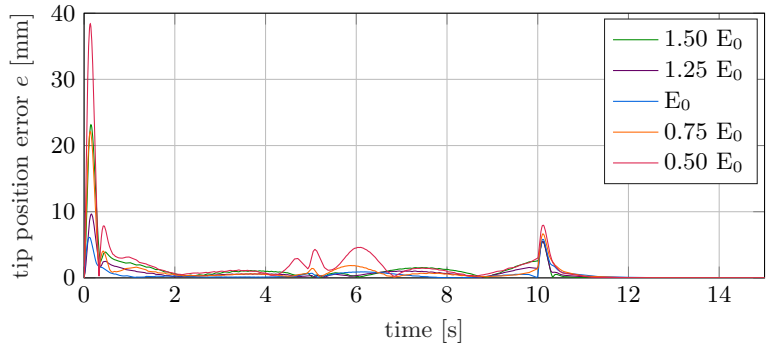
The trajectory tracking control results for the LQI controller are shown in Fig. 6.21. The x- and y-coordinate of the tip position as well as the tip position control error e are plotted over time for the different values of the Young's modulus E. In contrast to the other two controllers examined, the LQI controller shows no notable differences between trajectory tracking with the modified Young's modulus and the nominal Young's modulus of the simulation. Note that the controller gains are only determined for the nominal values. There are only small deviations at the beginning of the trajectory, which can be explained by the modified quasi-static behavior of the soft robot model. As a result, the controller initially determines the required system input associated with the equilibrium position of the robot at the current operating point with a comparatively large error. After a short time, however, the controller compensates for this error. The LQI controller has no difficulty with the changes that also occur in the dynamic behavior of the controlled robot. The reasons for the good controller performance are the integral component of the controller, the large gain of the controller, and the high sampling frequency of f = 1 kHz, which allows the controller to react quickly to changes.



a) x-coordinate.



b) z-coordinate.



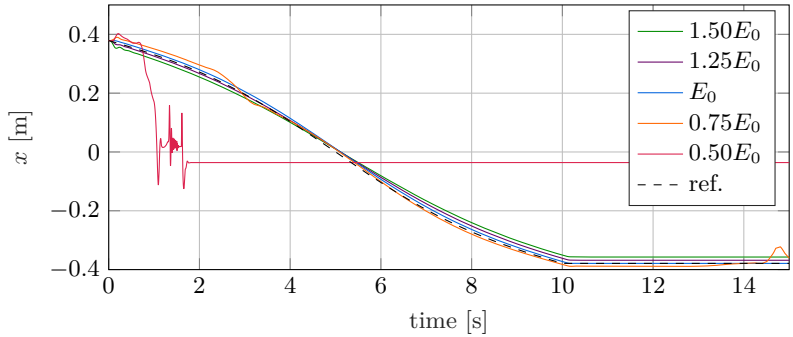
c) Error.

Figure 6.21: Trajectories of the tip position and control error for the LQI controller for different values of the Young's modulus E.

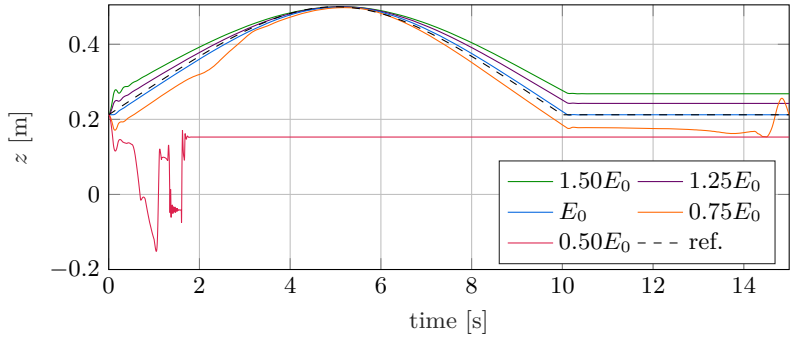
Model Predictive Controller

The trajectory tracking control results for the MPC are shown in Fig. 6.22. The x- and y-coordinate of the tip position as well as the tip position control error e are plotted over time for the different values of the Young's modulus E. The control error is shown only for those values of Young's modulus for which successful trajectory tracking was possible. Overall, for the MPC robust trajectory tracking control can be observed. However, the performance depends on the Young's modulus of the simulation model. For a larger Young's modulus of the simulation model, the z- coordinate of the tip position is always lager than the desired value. This corresponds to the curvatures of the soft robot being too small. This can be explained by the increased stiffness of the simulation model. The system input is determined by the MPC by solving an optimization problem in such a way that the desired motion is achieved with the controller-internal system model which, however, uses the nominal parameters. This results in an underestimation of the required tendon forces. If these system inputs are applied to the simulation model with the increased Young's modulus, the resulting curvature is lower than expected by the controller. The too low curvature can also be observed in the x coordinate of the robot tip.

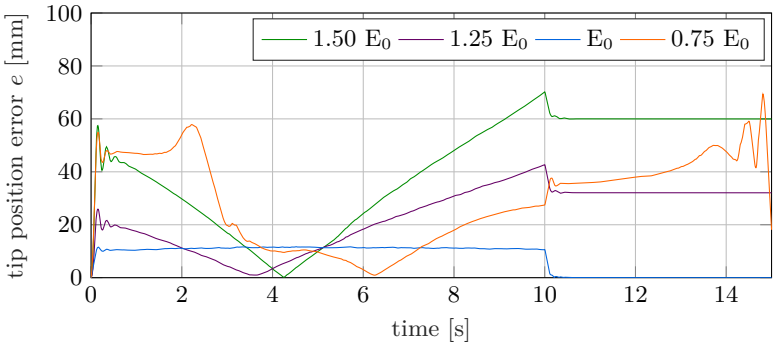
For the simulation model with a decreased Young's modulus of $0.75E_0$ the opposite effects can be observed, which, however, have the same reason. As the model integrated into the MPC is stiffer than the simulation model, the MPC overestimates the required tendon forces, and the resulting curvature of the simulation model is larger than desired. If the Young's modulus of the simulation model is further reduced to a value of $0.5E_0$, the MPC fails to find a meaningful solution and after some time simply keeps the system inputs constant as no improved value can be found. It can be observed that the MPC calculates ever-increasing system inputs to follow the reference trajectory with the internal model. This causes the curvature of the simulation model to increase more and more until the soft robot simply "curls up". This results in a totally different behavior of the simulation model and the model integrated in the MPC. From the time $t = 1.3\,\mathrm{s}$ on, the optimizer integrated in the MPC does not converge and is no longer able to find a solution to the optimization problem. From this moment on, the system inputs calculated by the MPC are kept constant at the last "successfully" determined value. Due to the internal damping, the soft robot oscillates to the corresponding equilibrium position and finally remains in this position.



a) x-coordinate.



b) z-coordinate.



c) Error.

Figure 6.22: Trajectories of the tip position and control error for the MPC for different values of the Young's modulus E.

6.6.4 Computation Time

For the application to physical systems, the computation time is important, as real-time capability is essential for feedback controllers. The computation time is very similar for all three trajectories. Therefore, only the computation time for the step trajectory is considered in the following. In Tab. 6.5 the sample time and the calculation time per control step are listed for all three controllers. It can be clearly seen that all three controllers are real-time capable. However, the computation time differs considerably.

The quasi-static controller is the fastest of the three controllers with an evaluation time of only 5.4 µs per iteration, which is several orders of magnitude less than the sample time of 200 ms. However, the sample time should not be reduced any further, as this will cause stability problems, as explained in Sec. 6.3. The LQI controller has a slightly higher computation time of 11.2 µs. This is almost two orders of magnitude faster than the chosen sample time of 1 ms for the LQI controller. The MPC is the only controller where computation time is a limiting factor. It takes 49.7 ms to compute one control step. For this reason, a sample time of 100 ms is chosen.

Table 6.5: Computation time and sample time of the three controllers for the step trajectory.

Controller	Sample Time	Calculation Time per Control Step
quasi-static	200 ms	5.4 µs
LQI	1 ms	11.2 µs
MPC	100 ms	49.7 ms

Conclusion

Modern fields of robotic applications, such as the collaborative and flexible use of robots in advanced manufacturing, the handling of fragile and differently shaped objects, and the interaction with unstructured environments in logistics, are becoming increasingly important. Here, it is not the overall strength and absolute accuracy of robots that is important, but the ability to interact safely, interactively and flexibly with humans and the environment. Soft robots are particularly strong in these areas. However, due to the complex dynamic behavior of soft robots, their use is typically limited to applications where slow motions can be accepted. For wider application in industry and logistics, soft robots that are both agile and accurate are needed. Therefore, efficient and accurate dynamic models, soft sensors for state estimation, and control algorithms for soft robots, which combined allow agile trajectory tracking control, are the subject of current research.

This work contributes to the efficient agile trajectory tracking control of soft robots. For this, methods for modeling, sensing and control for agile trajectory tracking of soft robots are investigated in simulation and experiment. The individual chapters of this work are summarized in the following.

First, in chapter 2 three different dynamic beam models known from soft robotics and multibody dynamics are presented and compared in terms of accuracy, computational efficiency and required discretization. All three models can be used to accurately model beam-shaped soft robots. The Cosserat rod and the absolute nodal coordinate formulation (ANCF) models provide the most accurate description of beam mechanics. The piecewise constant curvature (PCC) model is the most intuitive and simple model, but it is slightly less accurate because it does not include shear deformations, for example. Additionally, most existing dynamic formulations of the PCC model have issues with removable singularities in the straight configuration. For practical applications, however, the difference in accuracy is usually of minor importance as other unmodeled effects often play a much larger role. For typical soft robot parameters, the ANCF model is by far the slowest of the three models and by far not real-time capable, which limits its applicability to control applications. The Cosserat and the PCC are both real-time capable for typical soft robot designs, depending on the discretization. Finally, the integration of forces resulting from tendon actuation, which is one

of the most popular types of actuation for soft robots, into the beam models is demonstrated. The mathematical models presented here provide the basis for the design of the model-based controllers in the following chapters.

One of the main limitations of beam models for soft robotics applications is that non-ideal behavior, e.g. due to manufacturing inaccuracies, is difficult to include in these models. An alternative is the use of data-driven models, which are experimentally investigated in chapter 3. Here, a quasi-static and a dynamic data-driven model for soft robots are investigated. In a first step, a soft robot test system with pronounced dynamic behavior is introduced, which can be considered as a soft version of an inverted pendulum and is used as a test system throughout this work. In a next step, two data-driven models, a quasi-static model based on a simple feedforward neural network and a dynamic model based on spectral submanifold reduction (SSMR), are derived and investigated for the test system. It is shown that the quasi-static behavior of soft robots can be modeled with data-driven models in a data-efficient and accurate way. The dynamic behavior of soft robots can also be modeled with data-driven models in an accurate and computationally efficient way. However, compared to quasi-static modeling, more complex models and much more training data are required. This makes data-driven modeling of the dynamics of soft robots less efficient.

In addition to accurate models, sensor feedback is required for accurate control in real-world environments. In chapter 4, a planar and a spatial low-cost, easy to fabricate, fiber-optic curvature sensor that can be fully integrated into soft robots are presented and experimentally evaluated. The planar curvature sensor can measure the magnitude of bending in two spatial directions, but cannot determine the direction of bending. For the experimental evaluation of this sensor, three of these sensors are embedded in each of the three fingers of a three-finger gripper to enable shape sensing. It is shown that an accurate reconstruction of the bending of the fingers is possible for different types of grasps. The spatial curvature sensor can measure both magnitude and direction of curvature in two spatial directions. For experimental evaluation, the soft sensor is integrated into the soft robotic test system introduced in chapter 3. It is shown that with the spatial curvature sensor embedded in the body of the soft robot, an accurate estimation of the tip position of the soft robotic test system is possible.

In chapter 5, several approaches for feedforward trajectory tracking control are presented and experimentally evaluated with respect to their accuracy and computational efficiency. Specifically, a quasi-static data-driven, a dynamic model-based, and a novel hybrid control approach are presented. The latter one combines a quasi-static data-driven and a dynamic model-based part. It is shown that the quasi-static controller works very well for slow trajectories, but becomes less accurate for increasingly dynamic trajectories before finally failing to track the trajectory with an acceptable tolerance. The dynamic controller performs slightly worse for slow trajectories, but is able to follow fast trajectories

with reasonable accuracy. The hybrid controller outperforms the quasi-static and dynamic controllers for both slow and fast trajectories because it combines the most accurate quasi-static and dynamic models.

Finally, in chapter 6 different feedback control methods for dynamic trajectory tracking are presented. In a first step, a basic combined feedforward and feedback control approach is applied to the soft robotic test system introduced in chapter 3. For feedback, the spatial curvature sensor introduced in Sec. 4.2 is integrated into the soft robot. The data-driven quasi-static inverse model introduced in Sec. 5.1 is used as the feedforward part of the controller, and P and PI controllers are compared as the feedback part of the controller. Different types of distortions acting on the system are considered. It is shown that accurate trajectory tracking control is possible with this setup, but more advanced control concepts are needed for more agility. In a second step, more advanced feedback control approaches are presented and evaluated in simulation with respect to their accuracy, computational efficiency and robustness to parameter uncertainty. As a parameter uncertainty, variations of the Young's modulus by up to $\pm 50\,\%$ are considered. A quasi-static neural network based controller, a linear quadratic regulator with integral action (LQI) and gain scheduling and a model predictive controller (MPC) based on a PCC model are considered. The quasi-static controller performs the worst and is limited to slow trajectories. It is also the least robust of the three controllers and fails already at a slightly lower Young's modulus for the model. The LQI with gain scheduling and the MPC both follow dynamic trajectories accurately. The LQI with gain scheduling is the most robust to parameter uncertainty and is largely unaffected by a change in modulus. The MPC is also robust, but performs slightly worse and fails when the modulus is much lower than assumed by the controller.

With these aspects steps are taken towards enabling accurate and agile applications for soft robots. However, this work raises interesting further questions. The combination of the dynamic feedback controllers presented in chapter 6 using the spatial curvature sensor from chapter 4 with the hybrid feedforward controller from chapter 5 shall be investigated experimentally. Additionally, it shall also be investigated if the whole combined controller including the feedforward part is real-time capable to allow online trajectory tracking control.

In further research, the extension of the hybrid feedforward controller to a larger and more complex version of the soft robotic test system shall be investigated. For a larger version of the test system a finer discretization of the beam model is required. In a first step, this raises the question of the choice of the method for the derivation of the equations of motion for the forward model. For the coarse discretizations of the beam models considered in this work, a symbolic formulation is possible. For a finer discretization, however this is no longer possible due to the complexity of the equations of motions. Here other methods known from multibody dynamics such as the formulation of the equations of

motion as a DAE using redundant coordinates, the numerical calculation of the equations of motion at runtime in minimal coordinates, and the use of recursive formalisms to represent the equations of motion in minimal coordinates should be considered. Additionally, the properties of the resulting equations of motion of the forward model and of the inverse model obtained with the servo-constraints approach have to be investigated. With a finer discretization the equations of motion of the inverse model are no longer fully actuated. Therefore, internal dynamics exist for which stability has to be examined. Preliminary results show that stable internal dynamics can be expected for the PCC model.

In a second step, the incorporation of model uncertainties and parameter variations in the inverse model shall be investigated in more detail. This raises the question of which effects should be modeled with mechanical models and which effects are easier to model with data-driven approaches. The results of this work suggest that the quasi-static behavior is easier to model in a data-driven manner, while for modeling the dynamics, mechanical models tend to be more efficient. However, it is an open question whether a finer subdivision would bring further advantages. In addition, for the data-based model components, it should be investigated whether it makes more sense to derive them directly as an inverse model or to derive a forward model in a first step and invert it in a second step. For fully data-based, quasi-static, non-redundantly actuated models, it is common in soft robotics to derive the inverse model directly, as was done in this work. Compared to the construction of a forward model, which is inverted at runtime, there are great numerical advantages. However, it is an open question whether these advantages can be exploited in hybrid models. In addition, for hybrid models of redundant systems, it must be ensured that separately inverted model components select the same configuration of the robot.

Bibliography

[AbayazidEtAl13] Abayazid, M.; Kemp, M.; Misra, S.: 3D flexible needle steering in soft-tissue phantoms using Fiber Bragg Grating sensors. In 2013 IEEE international conference on robotics and automation, pp. 5843–5849, IEEE, 2013. DOI:10.1109/ICRA.2013.6631418.

[Adamy14] Adamy, J.: *Nichtlineare Systeme und Regelungen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. DOI:10.1007/978-3-642-45013-6.

[AKATJA GmbH23] AKATJA GmbH: DS20 Technical Specification, 2023.

[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 (RoboSoft), pp. 410–416, New Haven, CT, USA: IEEE, 2020. DOI:10.1109/RoboSoft48309.2020.9116015.

[AloraEtAl23] Alora, J.I.; Cenedese, M.; Schmerling, E.; Haller, G.; Pavone, M.: Data-Driven Spectral Submanifold Reduction for Nonlinear Optimal Control of High-Dimensional Robots. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 2627–2633, London, United Kingdom: IEEE, 2023. DOI:10.1109/ICRA48891.2023.10160418.

[Antman74] Antman, S.: Kirchhoff's problem for nonlinearly elastic rods. *Quarterly of Applied Mathematics*, Vol. 32, No. 3, pp. 221–240, 1974.

[ArmaniniEtAl23] Armanini, C.; Boyer, F.; Mathew, A.T.; Duriez, C.; Renda, F.: Soft Robots Modeling: A Structured Overview. *IEEE Transactions on Robotics*, Vol. 39, No. 3, pp. 1728–1748, 2023. DOI:10.1109/TRO.2022.3231360.

[BaillyAmirat05] Bailly, Y.; Amirat, Y.: Modeling and Control of a Hybrid Continuum Active Catheter for Aortic Aneurysm Treatment. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 924–929, Barcelona, Spain: IEEE, 2005. DOI:10.1109/ROBOT.2005.1570235.

[Bathe14] Bathe, K.J. (Ed.): *Finite element procedures*. Watertown, MA: K.J. Bathe, 2nd edition. Edn., 2014.

[BauchauTrainelli03] Bauchau, O.A.; Trainelli, L.: The Vectorial Parameterization of Rotation. *Nonlinear Dynamics*, Vol. 32, No. 1, pp. 71–92, 2003. DOI:10.1023/A:1024265401576.

[BestEtAl16] Best, C.M.; Gillespie, M.T.; Hyatt, P.; Rupert, L.; Sherrod, V.; Killpack, M.D.: A New Soft Robot Control Method: Using Model Predictive Control for a Pneumatically Actuated Humanoid. *IEEE Robotics & Automation Magazine*, Vol. 23, No. 3, pp. 75–84, 2016. DOI:10.1109/MRA.2016.2580591.

[BetschSteinmann02] Betsch, P.; Steinmann, P.: Frame-indifferent beam finite elements based upon the geometrically exact beam theory. *International Journal for Numerical Methods in Engineering*, Vol. 54, No. 12, pp. 1775–1788, 2002. DOI:10.1002/NME.487.

[BlackEtAl18] Black, C.B.; Till, J.; Rucker, D.C.: Parallel Continuum Robots: Modeling, Analysis, and Actuation-Based Force Sensing. *IEEE Transactions on Robotics*, Vol. 34, No. 1, pp. 29–47, 2018. DOI:10.1109/TRO.2017.2753829.

[BlajerKołodziejczyk04] Blajer, W.; Kołodziejczyk, K.: A Geometric Approach to Solving Problems of Control Constraints: Theory and a DAE Framework. *Multibody System Dynamics*, Vol. 11, No. 4, pp. 343–364, 2004. DOI:10.1023/B:MUBO.0000040800.40045.51.

[BobenkoSuris99] Bobenko, A.I.; Suris, Y.B.: Discrete Time Lagrangian Mechanics on Lie Groups, with an Application to the Lagrange Top. *Communications in Mathematical Physics*, Vol. 204, No. 1, pp. 147–188, 1999. DOI:10.1007/S002200050642.

[BruderEtAl21] Bruder, D.; Fu, X.; Gillespie, R.B.; Remy, C.D.; Vasudevan, R.: Data-Driven Control of Soft Robots Using Koopman Operator Theory. *IEEE Transactions on Robotics*, Vol. 37, No. 3, pp. 948–961, 2021. DOI:10.1109/TRO.2020.3038693.

[CamarilloEtAl09] Camarillo, D.; Carlson, C.; Salisbury, J.: Configuration Tracking for Continuum Manipulators With Coupled Tendon Drive. *IEEE Transactions on Robotics*, Vol. 25, No. 4, pp. 798–808, 2009. DOI:10.1109/TRO.2009.2022426.

[Campbell95] Campbell, S.L.: High-Index Differential Algebraic Equations. *Mechanics of Structures and Machines*, Vol. 23, No. 2, pp. 199–222, 1995. DOI:10.1080/08905459508905235.

[CenedeseEtAl22] Cenedese, M.; Axås, J.; Yang, H.; Eriten, M.; Haller, G.: Data-driven nonlinear model reduction to spectral submanifolds in mechanical systems. *Philosophical Transactions of the Royal Society A: Mathematical,*

Physical and Engineering Sciences, Vol. 380, No. 2229, p. 20210194, 2022. DOI:10.1098/RSTA.2021.0194.

[ChenEtAl24a] Chen, Z.; Ren, X.; Bernabei, M.; Mainardi, V.; Ciuti, G.; Stefanini, C.: A Hybrid Adaptive Controller for Soft Robot Interchangeability. *IEEE Robotics and Automation Letters*, Vol. 9, No. 1, pp. 875–882, 2024. DOI:10.1109/LRA.2023.3337705.

[ChenEtAl24b] Chen, Z.; Renda, F.; Gall, A.L.; Mocellin, L.; Bernabei, M.; Dangel, T.; Ciuti, G.; Cianchetti, M.; Stefanini, C.: Data-Driven Methods Applied to Soft Robot Modeling and Control: A Review. *IEEE Transactions on Automation Science and Engineering*, pp. 1–16, 2024. DOI:10.1109/TASE.2024.3377291.

[CianchettiEtAl12] Cianchetti, M.; Renda, F.; Licofonte, A.; Laschi, C.: Sensorization of continuum soft robots for reconstructing their spatial configuration. In 4th IEEE RAS & EMBS international conference on biomedical robotics and biomechatronics (BioRob), 2012, pp. 634–639, Piscataway, NJ: IEEE, 2012. DOI:10.1109/BIOROB.2012.6290788.

[ColemanEtAl93] Coleman, B.D.; Dill, E.H.; Lembo, M.; Lu, Z.; Tobias, I.: On the dynamics of rods in the theory of Kirchhoff and Clebsch. *Archive for Rational Mechanics and Analysis*, Vol. 121, No. 4, pp. 339–359, 1993. DOI:10.1007/BF00375625.

[CrisfieldJelenic99] Crisfield, M.A.; Jelenic, G.: Objectivity of strain measures in the geometrically exact three-dimensional beam theory and its finite-element implementation. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, Vol. 455, No. 1983, pp. 1125–1147, 1999.

[Della SantinaEtAl20a] Della Santina, C.; Bicchi, A.; Rus, D.: On an Improved State Parametrization for Soft Robots With Piecewise Constant Curvature and Its Use in Model Based Control. *IEEE Robotics and Automation Letters*, Vol. 5, No. 2, pp. 1001–1008, 2020. DOI:10.1109/LRA.2020.2967269.

[Della SantinaEtAl20b] 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*, Vol. 39, No. 4, pp. 490–513, 2020. DOI:10.1177/0278364919897292.

[Della SantinaEtAl23] Della Santina, C.; Duriez, C.; Rus, D.: Model-Based Control of Soft Robots: A Survey of the State of the Art and Open Challenges. *IEEE Control Systems*, Vol. 43, No. 3, pp. 30–65, 2023. DOI:10.1109/MCS.2023.3253419.

[DianEtAl22] Dian, S.; Zhu, Y.; Xiang, G.; Ma, C.; Liu, J.; Guo, R.: A Novel Disturbance-Rejection Control Framework for Cable-Driven Continuum Robots With Improved State Parameterizations. *IEEE Access*, Vol. 10, pp. 91545–91556, 2022. DOI:10.1109/ACCESS.2022.3202934.

[DrückerSeifried23] Drücker, S.; Seifried, R.: Trajectory-tracking control from a multibody system dynamics perspective. *Multibody System Dynamics*, Vol. 58, No. 3-4, pp. 341–363, 2023. DOI:10.1007/s11044-022-09870-9.

[Duske24] Duske, C.: *A data-based dynamic model of a soft robot for trajectory tracking control*. project thesis, Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, Hamburg, 2024.

[EngelEtAl05] Engel, Y.; Szabo, P.; Volkinshtein, D.: Learning to control an octopus arm with gaussian process temporal difference methods. *Advances in Neural Information Processing Systems*, Vol. 18, 2005.

[FalkenhahnEtAl15] Falkenhahn, V.; Hildebrandt, A.; Neumann, R.; Sawodny, O.: Model-based feedforward position control of constant curvature continuum robots using feedback linearization. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 762–767, Seattle, WA, USA: IEEE, 2015. DOI:10.1109/ICRA.2015.7139264.

[FalkenhahnEtAl17] Falkenhahn, V.; Hildebrandt, A.; Neumann, R.; Sawodny, O.: Dynamic Control of the Bionic Handling Assistant. *IEEE/ASME Transactions on Mechatronics*, Vol. 22, No. 1, pp. 6–17, 2017. DOI:10.1109/TMECH.2016.2605820.

[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. *IEEE/ASME Transactions on Mechatronics*, Vol. 27, No. 6, pp. 5296–5306, 2022. DOI:10.1109/TMECH.2022.3178303.

[FerreiraReis23] Ferreira, B.; Reis, J.: A Systematic Literature Review on the Application of Automation in Logistics. *Logistics*, Vol. 7, No. 4, p. 80, 2023. DOI:10.3390/logistics7040080.

[Franco22] Franco, E.: Model-Based Eversion Control of Soft Growing Robots With Pneumatic Actuation. *IEEE Control Systems Letters*, Vol. 6, pp. 2689–2694, 2022. DOI:10.1109/LCSYS.2022.3175385.

[GallowayEtAl19] Galloway, K.C.; Chen, Y.; Templeton, E.; Rife, B.; Godage, I.S.; Barth, E.J.: Fiber optic shape sensing for soft robotics. *Soft Robotics*, Vol. 6, No. 5, pp. 671–684, 2019.

[GaschEtAl21] Gasch, R.; Knothe, K.; Liebich, R.: *Strukturdynamik: Diskrete Systeme und Kontinua*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021. DOI:10.1007/978-3-662-61768-7.

[GeEtAl16a] Ge, J.; James, A.E.; Xu, L.; Chen, Y.; Kwok, K.W.; Fok, M.P.: Bidirectional soft silicone curvature sensor based on off-centered embedded fiber bragg grating. *IEEE Photonics Technology Letters*, Vol. 28, No. 20, pp. 2237–2240, 2016. DOI:10.1109/LPT.2016.2590984.

[GeEtAl16b] Ge, J.; James, A.E.; Xu, L.; Chen, Y.; Kwok, K.W.; Fok, M.P.: Bidirectional Soft Silicone Curvature Sensor Based on Off-Centered Embedded Fiber Bragg Grating. *IEEE Photonics Technology Letters*, Vol. 28, No. 20, pp. 2237–2240, 2016. DOI:10.1109/LPT.2016.2590984.

[GillespieEtAl18] Gillespie, M.T.; Best, C.M.; Townsend, E.C.; Wingate, D.; Killpack, M.D.: Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. In 2018 IEEE International Conference on Soft Robotics (RoboSoft), pp. 39–45, Livorno: IEEE, 2018. DOI:10.1109/ROBOSOFT.2018.8404894.

[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, Tokyo: IEEE, 2013. DOI:10.1109/IROS.2013.6697084.

[GrossEtAl12] Gross, D.; Hauger, W.; Schröder, J.; Wall, W.A.: *Technische Mechanik 2: Elastostatik*. Springer-Lehrbuch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. DOI:10.1007/978-3-642-19984-4.

[GrubeEtAl22] Grube, M.; Wieck, J.C.; Seifried, R.: Comparison of Modern Control Methods for Soft Robots. *Sensors*, Vol. 22, No. 23, p. 9464, 2022. DOI:10.3390/s22239464.

[GrubeEtAl24] Grube, M.; Drücker, S.; Seifried, R.: Open Loop Dynamic Trajectory Tracking Control of a Soft Robot using Learned Inverse Kinematics combined with a Dynamic Model. In 2024 European Control Conference (ECC), pp. 2972–2977, Stockholm, Sweden: IEEE, 2024. DOI:10.23919/ECC64448.2024.10591044.

[GrubeEtAl25] Grube, M.; Schumacher, C.M.; Seifried, R.: An Optical Curvature Sensor for Tip Position Trajectory Tracking Control of Soft Robots. In 2025 IEEE 8th International Conference on Soft Robotics (RoboSoft), pp. 1–6, Lausanne, Swizerland: IEEE, 2025.

[GrubeSeifried22a] Grube, M.; Seifried, R.: An Optical Curvature Sensor for Soft Robots. In A. Kecskeméthy; V. Parenti-Castelli (Eds.) ROMANSY 24 - Robot Design, Dynamics and Control, Vol. 606, pp. 125–132. Cham: Springer International Publishing, 2022.

[GrubeSeifried22b] Grube, M.; Seifried, R.: Simulation of soft robots with nonlinear material behavior using the cosserat rod theory. In 8th European

Congress on Computational Methods in Applied Sciences and Engineering, CIMNE, 2022. DOI:10.23967/ECCOMAS.2022.252.

[GrubeSeifried23] Grube, M.; Seifried, R.: An Optical Shape Sensor for Integration in Soft Grippers. In 2023 6th IEEE International Conference on Soft Robotics (RoboSoft), pp. 1–6, Singapore, Singapore: IEEE, 2023. DOI:10.1109/ROBOSOFT55895.2023.10122071.

[HallerPonsioen16] Haller, G.; Ponsioen, S.: Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction. *Nonlinear Dynamics*, Vol. 86, No. 3, pp. 1493–1534, 2016. DOI:10.1007/s11071-016-2974-z.

[Harte23] Harte, R.: *Spectral Mapping Theorems: A Bluffer's Guide*. Cham: Springer International Publishing, 2023. DOI:10.1007/978-3-031-13917-8.

[HendricksEtAl08] Hendricks, E.; Jannerup, O.; Haase Sørensen, P.: *Linear Systems Control: Deterministic and Stochastic Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. DOI:10.1007/978-3-540-78486-9.

[Hitec Commercial Solutions22] Hitec Commercial Solutions: HS-311 General Specification., 2022.

[Hui09] Hui, R.: *Fiber optic measurement techniques*. Amsterdam and London: Elsevier/Academic Press, 2009.

[Janabi-SharifiEtAl21] Janabi-Sharifi, F.; Jalali, A.; Walker, I.D.: Cosserat Rod-Based Dynamic Modeling of Tendon-Driven Continuum Robots: A Tutorial. *IEEE Access*, Vol. 9, pp. 68703–68719, 2021. DOI:10.1109/ACCESS.2021.3077186.

[JentoftHowe11] Jentoft, L.P.; Howe, R.D.: Compliant fingers make simple sensors smart. In International workshop on underactuated grasping 2010, pp. 114–121, Red Hook, NY: Curran, 2011.

[JonesWalker06] Jones, B.; Walker, I.: Kinematics for multisection continuum robots. *IEEE Transactions on Robotics*, Vol. 22, No. 1, pp. 43–55, 2006. DOI:10.1109/TRO.2005.861458.

[KapadiaEtAl10] Kapadia, A.D.; Walker, I.D.; Dawson, D.M.; Tatlicioglu, E.: A model-based sliding mode controller for extensible continuum robots. In Proceedings of the 9th WSEAS international conference on signal processing, robotics and automation, ISPRA'10, pp. 113–120, Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2010.

[KapadiaWalker11] Kapadia, A.D.; Walker, I.D.: Task-space control of extensible continuum manipulators. In 2011 IEEE/RSJ International Conference on

Intelligent Robots and Systems, pp. 1087–1092, San Francisco, CA: IEEE, 2011. DOI:10.1109/IROS.2011.6094873.

[KeshvarparastEtAl24] Keshvarparast, A.; Battini, D.; Battaia, O.; Pirayesh, A.: Collaborative robots in manufacturing and assembly systems: literature review and future research agenda. *Journal of Intelligent Manufacturing*, Vol. 35, No. 5, pp. 2065–2118, 2024. DOI:10.1007/s10845-023-02137-w.

[KhanEtAl20] Khan, A.H.; Shao, Z.; Li, S.; Wang, Q.; Guan, N.: Which is the best PID variant for pneumatic soft robots an experimental study. *IEEE/CAA Journal of Automatica Sinica*, Vol. 7, No. 2, pp. 451–460, 2020. DOI:10.1109/JAS.2020.1003045.

[KimEtAl21] Kim, D.; Kim, S.H.; Kim, T.; Kang, B.B.; Lee, M.; Park, W.; Ku, S.; Kim, D.; Kwon, J.; Lee, H.; Bae, J.; Park, Y.L.; Cho, K.J.; Jo, S.: Review of machine learning methods in soft robotics. *Plos One*, Vol. 16, No. 2, p. e0246102, 2021. DOI:10.1371/JOURNAL.PONE.0246102.

[KomenoEtAl22] Komeno, N.; Michael, B.; Kuchler, K.; Anarossi, E.; Matsubara, T.: Deep Koopman with Control: Spectral Analysis of Soft Robot Dynamics. In 2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE), pp. 333–340, Kumamoto, Japan: IEEE, 2022. DOI:10.23919/SICE56594.2022.9905758.

[KramerEtAl11] Kramer, R.K.; Majidi, C.; Sahai, R.; Wood, R.J.: Soft curvature sensors for joint angle proprioception. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1919–1926, San Francisco, CA: IEEE, 2011. DOI:10.1109/IROS.2011.6094701.

[KunkelMehrmann04] Kunkel, P.; Mehrmann, V.: Index reduction for differential-algebraic equations by minimal extension. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 84, No. 9, pp. 579–597, 2004. DOI:10.1002/ZAMM.200310127.

[LangEtAl11] Lang, H.; Linn, J.; Arnold, M.: Multi-body dynamics simulation of geometrically exact Cosserat rods. *Multibody System Dynamics*, Vol. 25, No. 3, pp. 285–312, 2011. DOI:10.1007/s11044-010-9223-x.

[LangLinn09] Lang, H.; Linn, J.: *Lagrangian field theory in space-time for geometrically exact Cosserat rods*, Vol. 3. ITWM, Fraunhofer Inst. Techno-und Wirtschaftsmathematik, 2009.

[LeeEtAl17a] Lee, C.; Kim, M.; Kim, Y.J.; Hong, N.; Ryu, S.; Kim, H.J.; Kim, S.: Soft robot review. *International Journal of Control, Automation and Systems*, Vol. 15, No. 1, pp. 3–15, 2017. DOI:10.1007/s12555-016-0462-3.

[LeeEtAl17b] Lee, K.H.; Fu, D.K.; Leong, M.C.; Chow, M.; Fu, H.C.; Althoefer, K.; Sze, K.Y.; Yeung, C.K.; Kwok, K.W.: Nonparametric Online

Learning Control for Soft Continuum Robot: An Enabling Technique for Effective Endoscopic Navigation. *Soft Robotics*, Vol. 4, No. 4, pp. 324–337, 2017. DOI:10.1089/SORO.2016.0065.

[LiEtAl17] Li, S.; Zhao, H.; Shepherd, R.F.: Flexible and stretchable sensors for fluidic elastomer actuated soft robots. *MRS Bulletin*, Vol. 42, No. 02, pp. 138–142, 2017. DOI:10.1557/MRS.2017.4.

[LiEtAl22] Li, Y.; Wang, X.; Kwok, K.W.: Towards Adaptive Continuous Control of Soft Robotic Manipulator using Reinforcement Learning. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7074–7081, Kyoto, Japan: IEEE, 2022. DOI:10.1109/IROS47612.2022.9981335.

[LinnEtAl13] Linn, J.; Lang, H.; Tuganov, A.: Geometrically exact Cosserat rods with Kelvin–Voigt type viscous damping. *Mechanical Sciences*, Vol. 4, No. 1, pp. 79–96, 2013. DOI:10.5194/MS-4-79-2013.

[LiuEtAl24] Liu, J.; Borja, P.; Della Santina, C.: Physics-Informed Neural Networks to Model and Control Robots: A Theoretical and Experimental Investigation. *Advanced Intelligent Systems*, Vol. 6, No. 5, p. 2300385, 2024. DOI:10.1002/AISY.202300385.

[López-GonzálezEtAl23] López-González, A.; Tejada, J.; López-Romero, J.: Review and Proposal for a Classification System of Soft Robots Inspired by Animal Morphology. *Biomimetics*, Vol. 8, No. 2, p. 192, 2023. DOI:10.3390/BIOMIMETICS8020192.

[MelinguiEtAl18] Melingui, A.; Mvogo Ahanda, J.J.B.; Lakhal, O.; Mbede, J.B.; Merzouki, R.: Adaptive Algorithms for Performance Improvement of a Class of Continuum Manipulators. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 48, No. 9, pp. 1531–1541, 2018. DOI:10.1109/TSMC.2017.2678605.

[NocedalWright06] Nocedal, J.; Wright, S.J.: *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2. Edn., 2006. DOI:10.1007/978-0-387-40065-5.

[Olson11] Olson, E.: AprilTag: A robust and flexible visual fiducial system. In 2011 IEEE International Conference on Robotics and Automation, pp. 3400–3407, Shanghai, China: IEEE, 2011. DOI:10.1109/ICRA.2011.5979561.

[OuyangEtAl18] Ouyang, B.; Mo, H.; Chen, H.; Liu, Y.; Sun, D.: Robust Model-Predictive Deformation Control of a Soft Object by Using a Flexible Continuum Robot. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 613–618, Madrid: IEEE, 2018. DOI:10.1109/IROS.2018.8593880.

[OzelEtAl15] Ozel, S.; Keskin, N.A.; Khea, D.; Onal, C.D.: A precise embedded curvature sensor module for soft-bodied robots. *Sensors and Actuators A: Physical*, Vol. 236, pp. 349–356, 2015. DOI:10.1016/J.SNA.2015.09.041.

[OzelEtAl16] Ozel, S.; Skorina, E.H.; Luo, M.; Tao, W.; Chen, F.; Pan, Y.; Onal, C.D.: A composite soft bending actuation module with integrated curvature sensing. In A. Okamura; A. Menciassi (Eds.) 2016 IEEE international conference on robotics and automation, stockholm, sweden, may 16th-21st, pp. 4963–4968, Piscataway, NJ: IEEE, 2016. DOI:10.1109/ICRA.2016.7487703.

[Pearson01] Pearson, K.: LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Vol. 2, No. 11, pp. 559–572, 1901. DOI:10.1080/14786440109462720.

[PolygerinosEtAl11] Polygerinos, P.; Ataollahi, A.; Schaeffter, T.; Razavi, R.; Seneviratne, L.D.; Althoefer, K.: MRI-compatible intensity-modulated force sensor for cardiac catheterization procedures. *IEEE Transactions on Biomedical Engineering*, Vol. 58, No. 3, pp. 721–726, 2011. DOI:10.1109/TBME.2010.2095853.

[PolygerinosEtAl17] Polygerinos, P.; Correll, N.; Morin, S.A.; Mosadegh, B.; Onal, C.D.; Petersen, K.; Cianchetti, M.; Tolley, M.T.; Shepherd, R.F.: Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing, control, and applications in human-robot interaction. *Advanced Engineering Materials*, Vol. 19, No. 12, p. 1700016, 2017. DOI:10.1002/ADEM.201700016.

[Porter71] Porter, B.: Optimal control of multivariable linear systems incorporating integral feedback. *Electronics Letters*, Vol. 7, No. 8, pp. 170–172, 1971. DOI:10.1049/EL:19710113.

[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*, Vol. 30, No. 5, pp. 1109–1122, 2014. DOI:10.1109/TRO.2014.2325992.

[RendaEtAl22] Renda, F.; Armanini, C.; Mathew, A.; Boyer, F.: Geometrically-Exact Inverse Kinematic Control of Soft Manipulators With General Threadlike Actuators' Routing. *IEEE Robotics and Automation Letters*, Vol. 7, No. 3, pp. 7311–7318, 2022. DOI:10.1109/LRA.2022.3183248.

[RichaletEtAl78] Richalet, J.; Rault, A.; Testud, J.; Papon, J.: Model predictive heuristic control. *Automatica*, Vol. 14, No. 5, pp. 413–428, 1978. DOI:10.1016/0005-1098(78)90001-8.

[Riek17] Riek, L.D.: Healthcare robotics. *Communications of the ACM*, Vol. 60, No. 11, pp. 68–78, 2017. DOI:10.1145/3127874.

[RobertsEtAl13] Roberts, P.; Damian, D.D.; Shan, W.; Lu, T.; Majidi, C.: Soft-matter capacitive sensor for measuring shear and pressure deformation. In 2013 IEEE international conference on robotics and automation, pp. 3529–3534, IEEE, 2013. DOI:10.1109/ICRA.2013.6631071.

[RoneBen-Tzvi14] Rone, W.S.; Ben-Tzvi, P.: Continuum Robot Dynamics Utilizing the Principle of Virtual Power. *IEEE Transactions on Robotics*, Vol. 30, No. 1, pp. 275–287, 2014. DOI:10.1109/TRO.2013.2281564.

[RuckerWebster III11] Rucker, D.C.; Webster III, R.J.: Statics and Dynamics of Continuum Robots With General Tendon Routing and External Loading. *IEEE Transactions on Robotics*, Vol. 27, No. 6, pp. 1033–1044, 2011. DOI:10.1109/TRO.2011.2160469.

[RusTolley15] Rus, D.; Tolley, M.T.: Design, fabrication and control of soft robots. *Nature*, Vol. 521, No. 7553, pp. 467–475, 2015. DOI:10.1038/NATURE14543.

[RyuEtAl12] Ryu, S.C.; Quek, Z.F.; Renaud, P.; Black, R.J.; Daniel, B.L.; Cutkosky, M.R.: An optical actuation system and curvature sensor for a MR-compatible active needle. In 2012 IEEE International Conference on Robotics and Automation, Vol. 2012, pp. 1589–1594, 2012. DOI:10.1109/ICRA.2012.6224964.

[Sander10] Sander, O.: Geodesic finite elements for Cosserat rods. *International Journal for Numerical Methods in Engineering*, Vol. 82, No. 13, pp. 1645–1670, 2010. DOI:10.1002/NME.2814.

[SatheeshbabuEtAl19] Satheeshbabu, S.; Uppalapati, N.K.; Chowdhary, G.; Krishnan, G.: Open Loop Position Control of Soft Continuum Arm Using Deep Reinforcement Learning. In 2019 International Conference on Robotics and Automation (ICRA), pp. 5133–5139, Montreal, QC, Canada: IEEE, 2019. DOI:10.1109/ICRA.2019.8793653.

[SauerEtAl91] Sauer, T.; Yorke, J.A.; Casdagli, M.: Embedology. *Journal of Statistical Physics*, Vol. 65, No. 3-4, pp. 579–616, 1991. DOI:10.1007/BF01053745.

[SchiehlenEberhard20] Schiehlen, W.; Eberhard, P.: *Technische Dynamik: Aktuelle Modellierungs- und Berechnungsmethoden auf einer gemeinsamen Basis.* Wiesbaden: Springer Fachmedien Wiesbaden, 2020. DOI:10.1007/978-3-658-31373-9.

[Shabana11] Shabana, A.A.: *Computational Continuum Mechanics.* Cambridge University Press, 2. Edn., 2011. DOI:10.1017/CBO9781139059992.

[ShampineReichelt97] Shampine, L.F.; Reichelt, M.W.: The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, Vol. 18, No. 1, pp. 1–22, 1997. DOI:10.1137/S1064827594276424.

[Shoemake85] Shoemake, K.: Animating rotation with quaternion curves. In Proceedings of the 12th annual conference on Computer graphics and interactive techniques - SIGGRAPH '85, pp. 245–254, San Francisco: ACM Press, 1985. DOI:10.1145/325334.325242.

[Siciliano16] Siciliano, B.; Khatib, O. (Eds.): *Springer Handbook of Robotics*. Springer Handbooks. Cham: Springer International Publishing, 2016. DOI:10.1007/978-3-319-32552-1.

[SilverEtAl14] Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M.: Deterministic policy gradient algorithms. In International conference on machine learning, pp. 387–395, Pmlr, 2014.

[Smooth-On24] Smooth-On: Dragon Skin™ Series, 2024.

[SpillmanTeschner07] Spillman, J.; Teschner, M.: CoRdE: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '07, pp. 63–72, San Diego, California: Eurographics Association, 2007.

[Steinmetz24] Steinmetz, N.: *Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2024. DOI:10.1007/978-3-662-68086-5.

[StölzleDella Santina24] Stölzle, M.; Della Santina, C.: Input-to-state stable coupled oscillator networks for closed-form model-based control in latent space. In A. Globerson; L. Mackey; D. Belgrave; A. Fan; U. Paquet; J. Tomczak; C. Zhang (Eds.) Advances in neural information processing systems, Vol. 37, pp. 82010–82059, Curran Associates, Inc., 2024.

[Takens81] Takens, F.: Detecting strange attractors in turbulence. In Dynamical Systems and Turbulence, Warwick 1980: Proceedings of a symposium held at the University of Warwick 1979/80, pp. 366–381, Berlin, Heidelberg: Springer Berlin Heidelberg, 1981.

[TanEtAl22] Tan, K.; Ji, Q.; Feng, L.; Torngren, M.: Shape estimation of a 3D printed soft sensor using multi-hypothesis extended kalman filter. *IEEE Robotics and Automation Letters*, Vol. 7, No. 3, pp. 8383–8390, 2022. DOI:10.1109/LRA.2022.3187832.

[TangEtAl22] Tang, Z.; Wang, P.; Xin, W.; Laschi, C.: Learning-Based Approach for a Soft Assistive Robotic Arm to Achieve Simultaneous Position and Force Control. *IEEE Robotics and Automation Letters*, Vol. 7, No. 3, pp. 8315–8322, 2022. DOI:10.1109/LRA.2022.3185786.

[ThuruthelEtAl16a] Thuruthel, T.G.; Falotico, E.; Cianchetti, M.; Laschi, C.: Learning Global Inverse Kinematics Solutions for a Continuum Robot. In V. Parenti-Castelli; W. Schiehlen (Eds.) ROMANSY 21 - Robot Design, Dynamics and Control, Vol. 569, pp. 47–54. Cham: Springer International Publishing, 2016. Series Title: CISM International Centre for Mechanical Sciences.

[ThuruthelEtAl16b] Thuruthel, T.G.; Falotico, E.; Cianchetti, M.; Renda, F.; Laschi, C.: Learning Global Inverse Statics Solution for a Redundant Soft Robot:. In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics, pp. 303–310, Lisbon, Portugal: SCITEPRESS - Science and and Technology Publications, 2016. DOI:10.5220/0005979403030310.

[ThuruthelEtAl17a] Thuruthel, T.G.; Falotico, E.; Manti, M.; Pratesi, A.; Cianchetti, M.; Laschi, C.: Learning Closed Loop Kinematic Controllers for Continuum Manipulators in Unstructured Environments. *Soft Robotics*, Vol. 4, No. 3, pp. 285–296, 2017. DOI:10.1089/SORO.2016.0051.

[ThuruthelEtAl17b] Thuruthel, T.G.; Falotico, E.; Renda, F.; Laschi, C.: Learning dynamic models for open loop predictive control of soft robotic manipulators. *Bioinspiration & Biomimetics*, Vol. 12, No. 6, p. 066003, 2017. DOI:10.1088/1748-3190/AA839F.

[ThuruthelEtAl18a] Thuruthel, T.G.; Ansari, Y.; Falotico, E.; Laschi, C.: Control Strategies for Soft Robotic Manipulators: A Survey. *Soft Robotics*, Vol. 5, No. 2, pp. 149–163, 2018. DOI:10.1089/SORO.2017.0007.

[ThuruthelEtAl18b] Thuruthel, T.G.; Falotico, E.; Manti, M.; Laschi, C.: Stable Open Loop Control of Soft Robotic Manipulators. *IEEE Robotics and Automation Letters*, Vol. 3, No. 2, pp. 1292–1298, 2018. DOI:10.1109/LRA.2018.2797241.

[ThuruthelEtAl19] Thuruthel, T.G.; Falotico, E.; Renda, F.; Laschi, C.: Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators. *IEEE Transactions on Robotics*, Vol. 35, No. 1, pp. 124–134, 2019. DOI:10.1109/TRO.2018.2878318.

[Timoshenko40] Timoshenko, S.: *Strength of Materials-Part 1.* New York: D. Van Nostrand Company, Inc., 2. Edn., 1940.

[TrivediEtAl08] Trivedi, D.; Lotfi, A.; Rahn, C.: Geometrically Exact Models for Soft Robotic Manipulators. *IEEE Transactions on Robotics*, Vol. 24, No. 4, pp. 773–780, 2008. DOI:10.1109/TRO.2008.924923.

[TrumicEtAl21] Trumic, M.; Della Santina, C.; Jovanovic, K.; Fagiolini, A.: Adaptive Control of Soft Robots Based on an Enhanced 3D Augmented Rigid Robot Matching. In 2021 American Control Confer-

ence (ACC), pp. 4991–4996, New Orleans, LA, USA: IEEE, 2021. DOI:10.23919/ACC50511.2021.9482817.

[WangEtAl16] Wang, H.; Zhang, R.; Chen, W.; Liang, X.; Pfeifer, R.: Shape detection algorithm for soft manipulator based on fiber bragg gratings. *IEEE/ASME Transactions on Mechatronics*, Vol. 21, No. 6, pp. 2977–2982, 2016. DOI:10.1109/TMECH.2016.2606491.

[WangEtAl21] Wang, X.; Li, Y.; Kwok, K.W.: A Survey for Machine Learning-Based Control of Continuum Robots. *Frontiers in Robotics and AI*, Vol. 8, p. 730330, 2021. DOI:10.3389/FROBT.2021.730330.

[WebsterJones10] Webster, R.J.; Jones, B.A.: Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review. *The International Journal of Robotics Research*, Vol. 29, No. 13, pp. 1661–1683, 2010. DOI:10.1177/0278364910368147.

[Wieck21] Wieck, J.C.: *Vergleich von Simulationsverfahren für Softroboter*. project thesis, Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, Hamburg, 2021.

[WieseEtAl19] Wiese, M.; Runge-Borchert, G.; Raatz, A.: Optimization of Neural Network Hyperparameters for Modeling of Soft Pneumatic Actuators. In G. Carbone; M. Ceccarelli; D. Pisla (Eds.) New Trends in Medical and Service Robotics, Vol. 65, pp. 199–206. Cham: Springer International Publishing, 2019. Series Title: Mechanisms and Machine Science.

[YaoZhu14] Yao, S.; Zhu, Y.: Wearable multifunctional sensors using printed stretchable conductors made of silver nanowires. *Nanoscale*, Vol. 6, No. 4, pp. 2345–2352, 2014. DOI:10.1039/C3NR05496A.

[YiEtAl10] Yi, J.; Zhu, X.; Shen, L.; Sun, B.; Jiang, L.: An orthogonal curvature fiber bragg grating sensor array for shape reconstruction. In K. Li; X. Li; S. Ma; G.W. Irwin (Eds.) Life system modeling and intelligent computing, pp. 25–31, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[YipCamarillo14] Yip, M.C.; Camarillo, D.B.: Model-Less Feedback Control of Continuum Manipulators in Constrained Environments. *IEEE Transactions on Robotics*, Vol. 30, No. 4, pp. 880–889, 2014. DOI:10.1109/TRO.2014.2309194.

[YipCamarillo16] Yip, M.C.; Camarillo, D.B.: Model-Less Hybrid Position/Force Control: A Minimalist Approach for Continuum Manipulators in Unknown, Constrained Environments. *IEEE Robotics and Automation Letters*, Vol. 1, No. 2, pp. 844–851, 2016. DOI:10.1109/LRA.2016.2526062.

[YoungWillems72] Young, P.C.; Willems, J.C.: An approach to the linear multivariable servomechanism problem†. *International Journal of Control*, Vol. 15, No. 5, pp. 961–979, 1972. DOI:10.1080/00207177208932211.

[YuEtAl19] Yu, B.; Fernández, J.D.G.; Tan, T.: Probabilistic Kinematic Model of a Robotic Catheter for 3D Position Control. *Soft Robotics*, Vol. 6, No. 2, pp. 184–194, 2019. DOI:10.1089/SORO.2018.0074.

[ZhangEtAl17] Zhang, H.; Cao, R.; Zilberstein, S.; Wu, F.; Chen, X.: Toward Effective Soft Robot Control via Reinforcement Learning. In Y. Huang; H. Wu; H. Liu; Z. Yin (Eds.) Intelligent Robotics and Applications, Vol. 10462, pp. 173–184. Cham: Springer International Publishing, 2017. Series Title: Lecture Notes in Computer Science.

[ZhangEtAl19] Zhang, Y.; Gao, J.; Yang, H.; Hao, L.: A novel hysteresis modelling method with improved generalization capability for pneumatic artificial muscles. *Smart Materials and Structures*, Vol. 28, No. 10, p. 105014, 2019. DOI:10.1088/1361-665X/AB3770.

[Zhermack SpA13] Zhermack SpA: HT 45 Transparent: Technical Data Sheet, 2013.

[ČeponEtAl09] Čepon, G.; Manin, L.; Boltežar, M.: Introduction of damping into the flexible multibody belt-drive model: A numerical and experimental investigation. *Journal of Sound and Vibration*, Vol. 324, No. 1-2, pp. 283–296, 2009. DOI:10.1016/J.JSV.2009.02.001.