sensors

Article

Comparison of Modern Control Methods for Soft Robots

Malte Grube *, Jan Christian Wieck and Robert Seifried

Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, 21073 Hamburg, Germany
* Correspondence: malte.grube@tuhh.de

Abstract: With the rise in new soft robotic applications, the control requirements increase. Therefore, precise control methods for soft robots are required. However, the dynamic control of soft robots, which is required for fast movements, is still an open topic and will be discussed here. In this contribution, one kinematic and two dynamic control methods for soft robots are examined. Thereby, an LQI controller with gain scheduling, which is new to soft robotic applications, and an MPC controller are presented. The controllers are compared in a simulation regarding their accuracy and robustness. Additionally, the required implementation effort and computational effort is examined. For this purpose, the trajectory tracking control of a simple soft robot is studied for different trajectories. The soft robot is beam-shaped and tendon-actuated. It is modeled using the piecewise constant curvature model, which is one of the most popular modeling techniques in soft robotics. In this paper, it is shown that all three controllers are able to follow the examined trajectories. However, the dynamic controllers show much higher accuracy and robustness than the kinematic controller. Nevertheless, it should be noted that the implementation and computational effort for the dynamic controllers is significantly higher. Therefore, kinematic controllers should be used if movements are slow and small oscillations can be accepted, while dynamic controllers should be used for faster movements with higher accuracy or robustness requirements.

Keywords: soft robotics; control; dynamic control; kinematic control; piecewise constant curvature

check for updates

Citation: Grube, M.; Wieck, J.C.; Seifried, R. Comparison of Modern Control Methods for Soft Robots. *Sensors* **2022**, *22*, 9464. https://doi.org/10.3390/s22239464

Academic Editors: Hamed Rahimi Nohooji and Amirreza Fahim Golestaneh

Received: 2 November 2022
Accepted: 28 November 2022
Published: 3 December 2022

1. Introduction

Soft material robots are an emerging and fast-growing field of research with potential applications in various areas. In contrast to conventional robots, which are usually fabricated out of high-stiffness materials such as steel, soft robots are mostly fabricated out of soft materials such as silicone or foam. The material stiffness of soft robots is often in the range of $10^4 \dots 10^9$ Pa, which is comparable to the stiffness of biological tissue [1]. Due to the use of soft materials, very large deformations often occur. These are usually used to allow movements of the soft robot without the use of separate rigid joints. Typical applications of soft robots are medical applications and all sorts of human–machine interactions [2].

For the practical application of soft robot, feedback control is often required. As soft robots usually have a very task-specific design, the control methods used are often very specific as well. In this contribution, the focus is mainly on task-space control for trajectory tracking. In Section 1.1, a short overview over existing control approaches is given.

Controller design often requires a model. One of the most popular modeling techniques for soft robots, which will also be used in this paper, is the piecewise constant curvature (PCC) model [3]. This model subdivides a soft robot into sections with constant curvature. The PCC model is described in more detail in Section 2.2.

In this paper, one so-called kinematic control method and two dynamic control methods for tip-position trajectory tracking control of a tendon-actuated soft robot are compared in simulation. For each of the controllers, the achievable accuracy for trajectory tracking, the robustness against parameter uncertainty, the implementation effort and the computation time are examined. The kinematic control method is a model-free closed-loop control approach based on [4,5]. The first examined dynamic control method is a linear quadratic control approach with integral action and gain scheduling, which is novel to soft robotics. The second dynamic control method is a model predictive control approach, which has already been used for the control of different soft robots in the literature. Both dynamic control methods are model-based and closed loop. A PCC model of the soft robot is used for the simulation and for the design of the two model-based controllers.

1.1. Related Work

There is a large amount of work related to the control of soft robots. In the following, a short overview is given and summarized in Table 1. The control methods used for soft robots can be divided into two categories: model-based and model-free control. Furthermore, a distinction can be made between kinematic control, where the dynamics of the soft robot are neglected, and dynamic control, where the dynamics of the soft robot are considered [6]. In soft robotics literature, the terms "static controller" and "kinematic controller" are often used interchangeably. Moreover, a distinction can be made between open-loop and closed-loop control [2].

Table 1. Comparison of state-of-the-art control approaches for soft robots.

Class	Study	Model	Control Approach	Type
kinematic, model-based	[7]	PCC	direct inversion of kinematics	open-loop
	[8,9]	PCC	differential inversion of kinematics	open-loop
	[10]	Cosserat rod	differential inversion of kinematics	open-loop
	[11]	PCC	inversion of kinematics by optimization	open-loop
	[12]	PCC	inversion of kinematics with Jacobian transpose approach	open-loop
kinematic, model-free	[13]	-	learning of inverse kinematics with NN; learning on model	open-loop
	[14]	-	learning of inverse kinematics with NN; learning on physical robot	open-loop
	[4,5,15]	-	learning of inverse kinematics with NN	closed-loop
	[16]	-	learning of inverse kinematics with multitask Gaussian Process (open-loop) + locally weighted projection regression (closed-loop)	closed-loop
	[17,18]	-	online learning of kinematic Jacobian by incrementally moving each actuator	open-loop
dynamic, model-based	[19]	PCC	sliding mode	closed-loop
	[20–23]	PCC	PD	closed-loop
	[24]	rigid multibody system	MPC	closed-loop
	[25]	linear model + online Jacobian update	MPC	closed-loop
	[26]	PCC	adaptive control	closed-loop
	[27]	port-Hamiltonian	energy-shaping	closed-loop
dynamic, model-free	[28–30]	-	reinforcement-learning with Markov decision process	closed-loop
	[31]	-	supervised learning with NN	closed-loop
	[32–34]	-	MPC with NN model of soft robot	closed-loop

1.1.1. Model-Based Kinematic Control

Model-based kinematic controllers are the most widely used controllers in soft robotics [6]. Most model-based controllers use the piecewise constant curvature (PCC) model [1]. This model is described in Section 2.2 in more detail. However, other models such as the Cosserat rod model are also used [6].

One approach for open-loop model-based kinematic control is to directly invert the forward kinematics, as, e.g., shown by [7]. For the soft robot considered in [7], the deformation depends linearly on the actuation variables. This allows the direct inversion. Since the system is underactuated, the Moore–Penrose pseudoinverse is used.

For more complex soft robots, the relationship between actuation variables and control variables is usually nonlinear and not unique. Therefore, the inverse kinematics cannot be calculated directly. A popular approach here is to use differential inverse kinematics, as, e.g., shown by [8–10]. Other approaches to obtain the inverse kinematics are to solve the inversion as an optimization problem [11] or to use an iterative Jacobian transpose approach [12].

1.1.2. Model-Free Kinematic Control

An alternative to model-based kinematic control is model-free kinematic control. Model-free approaches are especially popular for highly nonlinear and nonuniform systems that are difficult to model [6]. In model-free kinematic control, the mapping of actuation variables to control variables is usually learned by a neural network. Model-free kinematic controllers were first proposed in [13]. In that approach, the inverse kinematics of a fully actuated robot are directly learned with a neural network. The training data for the neural network are obtained from the forward kinematics in simulation. In similar approaches, as, e.g., [14], the training data are obtained experimentally from a physical robot.

For redundant soft robots, the relationship between actuation variables and control variables is usually not unique. To achieve a smooth movement, the current configuration of the soft robot has to be considered for the determination of the control variables. This is, e.g., carried out in [4,5,15]. As the current configuration of the soft robot is needed for control, this is a closed-loop control approach. In [16], a combination of open-loop and closed-loop control is used. For the open-loop controller, the kinematics are learned using the multitask Gaussian Process method [35]. The closed-loop control behavior is learned using locally weighted projection regression (LWPR).

Offline training, which is usually very time-consuming, can be fully avoided by using online learning techniques, as, e.g., presented in [17,18]. There, the kinematic Jacobian is determined online by incrementally moving each actuator. However, with this approach, only very low control frequencies can be archived.

1.1.3. Model-Based Dynamic Control

For faster movements and higher accuracy requirements, kinematic control is often not sufficient as the dynamics of the soft robot cannot be neglected anymore. This can be solved by using model-based dynamic control. Compared to kinematic control, dynamic control is much more computationally expensive [6]. Thereby, the PCC model is also one of the most widely used models for model-based dynamic control of soft robots.

One of the first model-based dynamic controllers for soft robots was proposed in [19]. The soft robot is modeled with the PCC model, and the controller is a sliding mode controller. In [20], a closed-loop PD controller based on the same kinematic and dynamic model is presented. Similar control approaches for slightly different soft robots are presented in [21–23].

Another very popular control approach that has been successfully applied to soft robots is model predictive control. Model predictive controllers (MPCs) are usually computationally very expensive since, in every control step, the control output is obtained by solving an optimization problem. Therefore, mostly comparatively simple dynamic models of the soft robot are used. In [24], an MPC for an inflatable soft robot is presented. However, in the model used, the soft robot is assumed to be rigid. In [25], a model predictive control approach for soft robots is presented using a linear model with an online update of the Jacobian.

Another approach from classic nonlinear control is adaptive control. Adaptive control is robust to model uncertainties. This is important for in robotics, as, often, model parameters can only be approximated. It is used in [26] for the control of a multi-segment soft robot in 3D. One disadvantage of the formulation of the controller used in [26] is that it can only be used for fully actuated systems. However, soft robots are often underactuated. Finally, in [27], a control law constructed with an energy-shaping approach for a soft robot modeled as a port-Hamiltonian system is presented.

1.1.4. Model-Free Dynamic Control

Model-free dynamic controllers are especially used if no model of the soft robot is available or the available models are too slow for real-time control applications. This is often the case if soft robots are very complex. However, they strongly depend on the availability of training data.

Most model-free dynamic controllers are based on machine learning. In [28–30], different reinforcement learning approaches are presented based on a description of the control problem as a Markov decision process. For reinforcement learning, the controllers are usually pretrained in simulation, then the training is continued on the physical robot. A supervised learning approach based on neural networks is presented in [31]. In these approaches, the control output is directly obtained by machine learning.

An alternative approach is to use control concepts known from model-based control but use machine learning techniques to represent the model of the soft robot. This is, e.g., conducted for model predictive controllers in [32–34] for different soft robots.

2. Modeling

In this contribution, a tendon-driven soft robot arm is considered as an application example (see Figure 1). In the following, the used simulation model is presented. First, the design of the model is discussed. Then, the basic piecewise constant curvature model is presented to describe the soft robot. Finally, the inclusion of the cable actuation is discussed.

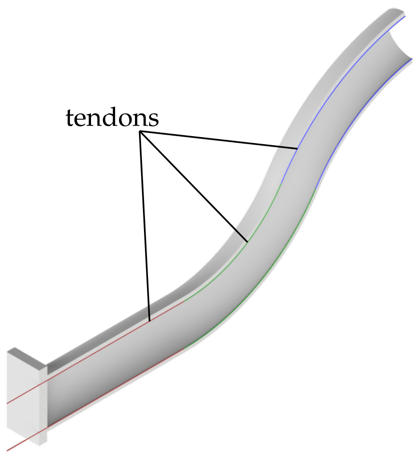

Figure 1. Cut through the soft robot.

2.1. Simulation Model of the Used Soft Robot

For the comparison of the controllers in this contribution, the control of a simple tendon-actuated soft robot, as shown in Figure 1, is examined in simulation. The model is based on a soft robot of long, slender shape, whose cross-section forms a circular ring with an outer radius of $R = 50\,\text{mm}$ and an inner radius of $r = 25\,\text{mm}$. The soft robot has a total length of $L_{\text{total}} = 500\,\text{mm}$ and a Young's modulus of $E = 7.32 \times 10^5\,\text{Pa}$. In this contribution, stiffness proportional damping is assumed with a damping constant of $E\mu = 36.6 \times 10^3\,\text{Nms}$. For the actuation, three pairs of tendons run along the top and bottom of the soft robot. The tendons have a distance of $r_{\text{cable}} = 25\,\text{mm}$ to the neutral fiber. These geometric and material properties are summarized in Table 2. For simplicity, only movements in the xz-plane are considered and modeled. However, this can be extended to 3D in a straight forward way. In the following, only the discretized version of the soft robot is described in more detail.

For the discretization, a PCC model with six segments with constant curvature is used. In Section 2.2, the PCC model is described in more detail. The discretized soft robot is shown schematically in Figure 2. For the complete description of the robot configuration, the curvatures β_i of the individual segments in the xz-plane are sufficient. However, for the full dynamical model, the rate of change of the curvature $\dot{\beta}_i$ is also required. The model is actuated by three tendon pairs, which run along the outside of the robot. The first tendon pair (indicated in red) ends at the second disk, the second tendon pair (indicated in green) ends at the fourth disk and the third tendon pair (indicated in blue) ends at the sixth disk. The tendon forces of the three tendon pairs are the control variables u_1, u_2, u_3 of the soft robot and can be summarized to the vector of control variables $\boldsymbol{u} = [u_1\, u_2\, u_3]^\mathrm{T}$.

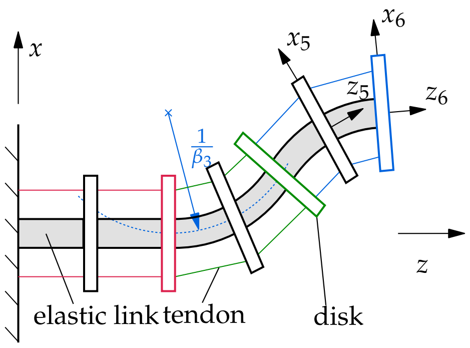

Figure 2. Schematic representation of the 2D system with 6 segments described as the PCC model.

Because the soft robot is assumed to be inextensible, only the force difference between the upper and lower tendon of each tendon pair is relevant. As tendons can only transmit pulling forces, it makes sense to assume that for each tendon pair one tendon transmits no force while the other one transmits the required pulling force. Therefore, only one control variable is needed for each tendon pair. A positive sign indicates that the force has to be applied at the upper tendon; a negative sign indicates that the force has to be applied at the lower tendon. In this contribution, it is assumed that all states β_i and $\dot{\beta}_i$ are accessible. 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 [36–39]. 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 [39], the rate of change $\dot{\beta}_i$ of the curvature can, e.g., be determined by numerical differentiation with good accuracy.

Table 2. 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_cable	distance of cables to centerline	25 mm
E	Young's modulus	7.32×10^5 Pa
$E\mu$	damping constant	36.6×10^3 Pas

2.2. Piecewise Constant Curvature Model

The piecewise constant curvature (PCC) model is one of the most popular modeling techniques for soft robots. It uses a state-space model to describe the soft robot. The method is presented by [3,40], among others. Compared to other techniques, such as the Cosserat rod theory, it uses fewer modeled degrees of freedom to represent a soft robot. This makes the model less complex than other methods. With a sufficiently fine discretization, it can still reflect the kinematics and dynamics of a soft robot sufficiently well.

The PCC model discretizes a soft robot by a series of N segments with constant curvature. In Figure 2, the discretization of a tendon-actuated soft robot with 6 segments is shown. Each of these segments consists of a disk, which is connected to the disk of the previous segment by an elastic link. The inertias and masses of the continuous soft robot are lumped in the disks. The properties of the elastic links are chosen to reflect the elasticity and damping of the real system. According to the model assumption, the links are massless, homogeneous cylinders with constant cross-sections, which have a constant elastic modulus E and constant shear modulus. The states of this model are the curvatures β_i of the sections as well as their derivatives $\dot{\beta}_i$. This modeling approach is able to represent bending deformations and, in the three-dimensional case, also torsion. Deformations due to shear or strain cannot be represented by this method. However, their influence on the total deformation of the robot is often small compared to bending and torsion and therefore can often be neglected. One advantage of using this model in control is that the curvature of the sections can be directly measured with a suitable sensor on the real robot, while the derivatives of the curvature can be obtained by numerical differentiation [39]. Therefore, in contrast to other methods, there is no need for an advanced state observer.

Following [40], the position $\boldsymbol{p}_{i,\text{local}}$ of disk i relative to the position of the previous disk $i-1$ can be described in a local coordinate frame using the curvatures β_i as

$$\boldsymbol{p}_{i,\text{local}} = \begin{bmatrix} \frac{1-\cos(\beta_i \ell_i)}{\beta_i} & 0 & \frac{\sin(\beta_i \ell_i)}{\beta_i} \end{bmatrix}^{\mathrm{T}}, \tag{1}$$

where ℓ_i is the length of segment i. The global location vector $\boldsymbol{p}_i$ of the i-th segment results from the location vector of the previous segment and its rotation matrix $\boldsymbol{R}_{i-1}$ to

$$\boldsymbol{p}_i = \begin{cases} \boldsymbol{p}_{i,\text{local}} & i = 1 \\ \boldsymbol{p}_{i-1} + \boldsymbol{R}_{i-1}\boldsymbol{p}_{i,\text{local}} & i > 1 \end{cases}. \tag{2}$$

The rotation matrix between the local coordinate systems of two neighboring segments is obtained in the two-dimensional case by a simple rotation by the angle $\theta_i = \beta_i \ell_i$ along the y-axis. By concatenating these rotations, the rotation matrix from the local coordinate system of the i-th disk to the reference system results in

$$\boldsymbol{R}_i = \begin{cases} \boldsymbol{R}_{i,\text{local}} & i = 1 \\ \boldsymbol{R}_{i-1}\boldsymbol{R}_{i,\text{local}} & i > 1 \end{cases}. \tag{3}$$

The equation of motion also requires the forces $\boldsymbol{f}_i$ and torques $\boldsymbol{\ell}_i$ acting on each segment. These act at the center of mass of the disk belonging to each segment. The force vector is calculated as

$$\boldsymbol{f}_i = \hat{\boldsymbol{f}}_i + \boldsymbol{f}_{i,\text{actuation}}, \tag{4}$$

where $\hat{\boldsymbol{f}}_i$ are external forces and $\boldsymbol{f}_{i,\text{actuation}}$ are forces resulting from the actuation. The torque acting on disk i includes the bending torques $\boldsymbol{\ell}_{i,\text{bnd}}$ and $\boldsymbol{\ell}_{i+1,\text{bnd}}$, the damping torques $\boldsymbol{\ell}_{i,\text{dmp}}$ and $\boldsymbol{\ell}_{i+1,\text{dmp}}$, the sum of the external torques $\hat{\boldsymbol{\ell}}_i$ and the torques due to actuation $\boldsymbol{\ell}_{i,\text{actuation}}$. This results in

$$\boldsymbol{\ell}_i = \begin{cases} \boldsymbol{\ell}_{i,\text{bnd}} - \boldsymbol{\ell}_{i+1,\text{bnd}} + \boldsymbol{\ell}_{i,\text{dmp}} - \boldsymbol{\ell}_{i+1,\text{dmp}} + \hat{\boldsymbol{\ell}}_i + \boldsymbol{\ell}_{i,\text{actuation}} & i < N \\ \boldsymbol{\ell}_{i,\text{bnd}} + \boldsymbol{\ell}_{i,\text{dmp}} + \hat{\boldsymbol{\ell}}_i + \boldsymbol{\ell}_{i,\text{actuation}} & i = N \end{cases}. \tag{5}$$

Under the assumption of linear-elastic material behavior, the bending torques can be calculated as

$$\boldsymbol{\ell}_{i,\text{bnd}} = -EJ_{yy}\beta_i \boldsymbol{e}_y, \tag{6}$$

where J_{yy} is the second moment of area around the y-axis, and $\boldsymbol{e}_y$ is the unit vector along the y-axis. In an analogous way, the damping torques can be determined under the assumption of stiffness-proportional viscous damping to

$$\boldsymbol{\ell}_{i,\mathrm{dmp}} = -E\mu J_{yy}\dot{\beta}_i \boldsymbol{e}_y. \tag{7}$$

Here, the product $E\mu$ is the damping constant. The derivation of the actuation forces $\boldsymbol{f}_{i,\mathrm{actuation}}$ and actuation torques $\boldsymbol{\ell}_{i,\mathrm{actuation}}$ for a tendon-actuated soft robot is described in Section 2.3.

With the mass m_i and inertia tensor J_i for all disks, the balance of linear and angular momenta can be established for all disks. Using the direct kinematics and the derived forces f_i and torques ℓ_i, from this follows the equation of motion of the form:

$$\boldsymbol{M}(\bar{\boldsymbol{x}})\ddot{\bar{\boldsymbol{x}}} = \boldsymbol{h}(\bar{\boldsymbol{x}}, \dot{\bar{\boldsymbol{x}}}, t). \tag{8}$$

Here, $\boldsymbol{M}(\bar{\boldsymbol{x}})$ is the mass matrix, and $\bar{\boldsymbol{x}} = \begin{bmatrix}\beta_1 & \beta_2 & \dots & \beta_N\end{bmatrix}^{\mathrm{T}}$ is the vector of generalized coordinates. With $\boldsymbol{x} = \begin{bmatrix}\bar{\boldsymbol{x}} & \dot{\bar{\boldsymbol{x}}}\end{bmatrix}^{\mathrm{T}}$, this can also be written as a state-space model:

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix}\dot{\bar{\boldsymbol{x}}} \\ \boldsymbol{M}^{-1}\boldsymbol{h}\end{bmatrix}}_{\boldsymbol{f}(t,\boldsymbol{x})}. \tag{9}$$

As an output function,

$$\boldsymbol{y} = \underbrace{\begin{bmatrix}1 & 0 & 0 \\ 0 & 0 & 1\end{bmatrix} \cdot \boldsymbol{p}_N(\boldsymbol{x})}_{\boldsymbol{g}(\boldsymbol{x})} \tag{10}$$

is chosen, which returns the tip position. However, depending on the control problem, different choices are possible. The entire nonlinear differential equation can thus be written as

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}(t,\boldsymbol{x}), \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}).\end{aligned} \tag{11}$$

2.3. Actuation Forces from Tendon Actuation

Besides pneumatic actuation, tendon actuation, which is used in this contribution, is one of the most popular actuation methods for soft robots. As tendons can only transmit pulling forces, in the planar case, pairs of two tendons have to be used to allow bending the robot in both directions. Thereby, one tendon is placed at the upper side of the robot and the other one at the lower side of the robot. The tendon configuration considered in this work is described in Section 2.1 in more detail.

For the calculation of the actuation forces, friction in the tendon guidance is neglected. Therefore, for each tendon, the tendon force F_q is constant over the whole tendon length. Here, the index q describes the position of the tendon in the tendon pair, where the upper tendon of the tendon pair has the index 1, and the lower tendon has the index 2. Each tendon pair can be considered separately. Therefore, in the following, only one tendon pair is considered. If, as in this contribution, more than one tendon pair is used, the forces and torques of the different tendon pairs can simply be summed up. A tendon pair can either pass through a disk, end at the disk or not reach this disk. It is assumed that the tendon pair passes the first $k-1$ disks and ends at disk k. Obviously, the largest actuation forces act on the disk where the tendon pair ends, and no actuation forces act on disks that are not reached. Note that actuation forces also act on all disks that the tendon pair only passes.

For the calculation of the actuation forces, the routing points of the tendons through the disks are of importance. These are shown in Figure 3 for the configuration of tendons used in this contribution. The locations of the routing points in relation to the center of gravity S_i of a disk are given by

$$\boldsymbol{r}_{i,1,\text{local}} = r_{\text{tendon}} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{12}$$

$$\boldsymbol{r}_{i,2,\text{local}} = r_{\text{tendon}} \begin{bmatrix} -1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}. \tag{13}$$

In global coordinates, these can be written as

$$\boldsymbol{r}_{i,1} = \boldsymbol{p}_i + \boldsymbol{R}_i \boldsymbol{r}_{i,1,\text{local}}, \tag{14}$$

$$\boldsymbol{r}_{i,2} = \boldsymbol{p}_i + \boldsymbol{R}_i \boldsymbol{r}_{i,2,\text{local}}. \tag{15}$$

From the position of the routing points in global coordinates, the normalized vector $\boldsymbol{c}_{q,i}$ can now be obtained. This vector describes the direction of the tendons—and thus the direction of the tendon forces—from disk i to disk $i-1$. For the i-th disk, they are computed by

$$\boldsymbol{c}_{q,i} = \frac{\boldsymbol{r}_{i-1,q} - \boldsymbol{r}_{i,q}}{\|\boldsymbol{r}_{i-1,q} - \boldsymbol{r}_{i,q}\|}. \tag{16}$$

The forces $\boldsymbol{f}_{\text{tendon},i}$ acting on disk i can now be calculated as

$$\boldsymbol{f}_{\text{tendon},i} = \begin{cases} \sum\limits_{q=1}^{2} \boldsymbol{c}_{q,i} F_q - \boldsymbol{c}_{q,i+1} F_q & i < k \\ \sum\limits_{q=1}^{2} \boldsymbol{c}_{q,i} F_q & i = k \\ \boldsymbol{0} & \text{otherwise} \end{cases}. \tag{17}$$

In an analogous way, the resulting torques $\boldsymbol{\ell}_{\text{tendon},i}$

$$\boldsymbol{\ell}_{\text{tendon},i} = \begin{cases} \sum\limits_{q=1}^{2} \left(\left(\boldsymbol{R}_i \cdot \boldsymbol{r}_{i,q,\text{local}} \right) \times \left(\boldsymbol{c}_{q,i} F_q - \boldsymbol{c}_{q,i+1} F_q \right) \right) & i < k \\ \sum\limits_{q=1}^{2} \left(\left(\boldsymbol{R}_i \cdot \boldsymbol{r}_{i,q,\text{local}} \right) \times \left(-\boldsymbol{c}_{q,i} F_q \right) \right) & i = k \\ \boldsymbol{0} & \text{otherwise} \end{cases} \tag{18}$$

can also be obtained.

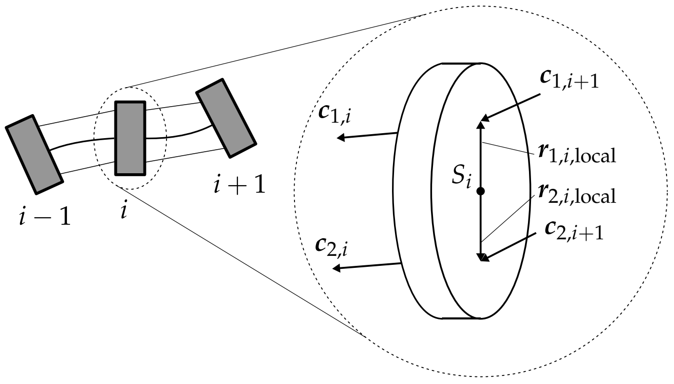

Figure 3. Position of tendon routing points and tendon force vectors on a disk.

3. Control Methods

In the following, the three examined control methods are briefly presented.

3.1. Kinematic Control

The first considered controller uses a model-free, kinematic approach based on [4,5]. Here, a shallow neural network is used to learn the global inverse kinematics of a soft robot. This can then be used to determine the necessary control values for a desired configuration of the robot. The dynamics of the system are neglected.

In general, the direct kinematics of the soft robot can be denoted as

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{u}) \tag{19}$$

with the vector of the control variables $\boldsymbol{u}$ and the resulting steady-state tip position $\boldsymbol{y}$. To determine the inverse kinematics, the direct kinematics from Equation (19) have to be inverted. Since the considered soft robot is kinematically redundant, there exists no unique global solution for this; the function $\boldsymbol{h}$ is surjective. However, the inverse differential kinematics

$$\delta\boldsymbol{y} = \boldsymbol{J}(\boldsymbol{u})\delta\boldsymbol{u} \tag{20}$$

can be used to obtain a locally unique solution [41]. Here, $\boldsymbol{J}(\boldsymbol{u})$ is the Jacobian-matrix of $\boldsymbol{h}(\boldsymbol{u})$ with respect to the control variables $\boldsymbol{u}$. The variations of the tip position $\boldsymbol{y}$ and the control variables $\boldsymbol{u}$ are denoted $\delta\boldsymbol{y}$ and $\delta\boldsymbol{u}$, respectively. Following [5], Equation (20) can be discretized such that

$$\boldsymbol{y}_{j+1} - \boldsymbol{y}_j = \boldsymbol{J}(\boldsymbol{u}_j)(\boldsymbol{u}_{j+1} - \boldsymbol{u}_j). \tag{21}$$

Here, $\boldsymbol{u}_{j+1}$ is the vector of control variables that reaches the positions $\boldsymbol{y}_{j+1}$ starting from the current position $\boldsymbol{y}_j$ with the corresponding control variables $\boldsymbol{u}_j$. In general, Equation (21) only holds if the difference between the control variables $\boldsymbol{u}_j$ and $\boldsymbol{u}_{j+1}$ is infinitesimally small. In practice, [41] shows that the equation also provides usable approximate solutions for larger distances.

For the kinematic control, Equation (21) must be solved for the control variable vector $\boldsymbol{u}_{j+1}$. This results in

$$\boldsymbol{u}_{j+1} = \boldsymbol{J}(\boldsymbol{u}_j)^{\dagger}(\boldsymbol{y}_{j+1} - \boldsymbol{y}_j + \boldsymbol{J}(\boldsymbol{u}_j)\boldsymbol{u}_j) = \boldsymbol{z}(\boldsymbol{y}_j, \boldsymbol{y}_{j+1}, \boldsymbol{u}_j) \tag{22}$$

with $\boldsymbol{J}(\boldsymbol{u}_j)^{\dagger}$ as the Moore–Penrose pseudoinverse of the Jacobian matrix $\boldsymbol{J}$. The mapping function of $\boldsymbol{y}_j$, $\boldsymbol{y}_{j+1}$ and $\boldsymbol{u}_j$ to $\boldsymbol{u}_{j+1}$ is denoted as $\boldsymbol{z}$ in the following. It represents a locally valid inversion of the kinematics of the soft robot. For the soft robot model considered in this contribution, the function of the local inverse kinematics $\boldsymbol{z}$ cannot be determined analytically. Therefore, in the following, it is approximated with a neural network with one hidden layer. The structure of the neural network is shown in Figure 4. The input layer has seven neurons. The hidden layer has 30 neurons and uses the tangent hyperbolicus as activation function. The output layer has three neurons and uses a linear activation function. The training is performed with Bayesian regularization.

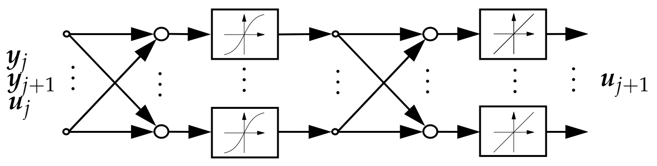

Figure 4. Structure of the neural network of the kinematic controller.

For the training of the neural network, a sufficient number of value pairs of the mapping $(\boldsymbol{y}_j, \boldsymbol{y}_{j+1}, \boldsymbol{u}_j) \rightarrow \boldsymbol{u}_{j+1}$ are required. To determine the required training data, the simulation model described in Section 2.1 is used. For the generation of one pair of values, the vector of control variables $\boldsymbol{u}_j$ and the vector of varied control variables $\boldsymbol{u}_{j+1}$ are chosen randomly such that $\boldsymbol{u}_j \in [-N_{\max}, N_{\max}]$ with $N_{\max} = 20\,\mathrm{N}$ and

$$|\boldsymbol{u}_j - \boldsymbol{u}_{j+1}| < \varepsilon N_{\max} \tag{23}$$

with $\varepsilon = 5\%$. Then, the robot is excited with the vector of control variables $\boldsymbol{u}_j$, and the resulting initial position $\boldsymbol{y}_j$ is determined after the transient processes have decayed. Subsequently, the control variables are changed to the new vector of control variables $\boldsymbol{u}_{j+1}$, which results in the corresponding position $\boldsymbol{y}_{j+1}$. In this way, a complete pair of values of the mapping $(\boldsymbol{y}_j, \boldsymbol{y}_{j+1}, \boldsymbol{u}_j) \to \boldsymbol{u}_{j+1}$ is determined. This procedure is repeated 10,000 times to collect the training data for the neural network. This takes about 5 min on a PC with an "Intel Core i7 - 6700K" processor. The resulting workspace of the soft robot is shown in Figure 5 and is reasonable and realistic for this type of robot.

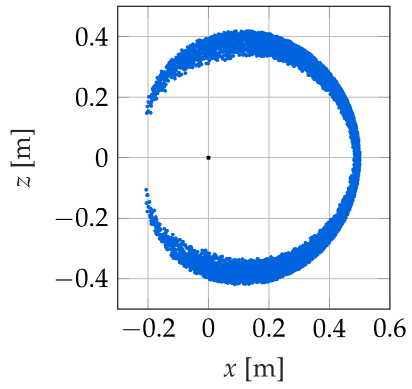

Figure 5. Empirically determined workspace of the kinematic controller for $N_{\max} = 20\,\mathrm{N}$.

The trained neural network can now be used as a controller for the soft robot, as shown in Figure 6. The inputs of the neural network are the current tip position $\boldsymbol{y}_j$, the current control variables $\boldsymbol{u}_j$ and the desired tip position $\boldsymbol{y}_{\mathrm{ref}}$. The outputs of the neural network are the control variables $\boldsymbol{u}_{j+1}$ that are required to archive the desired tip position. The controller runs with a sample frequency of $f_{\mathrm{controller}} = 5\,\mathrm{Hz}$. The reason for this comparatively low frequency is the kinematic behavior of the controller neglecting the dynamics of the soft robot. With each change in the control output $\boldsymbol{u}$, the dynamics of the system are excited; however, the kinematic controller assumes that the steady-state position is reached instantaneously. With the low sample frequency, the transients can at least partly decay before the measurements for the next control step are taken, which improves the performance and the stability of the controller. On the other hand, if the sample frequency of the controller is chosen even lower, the robot can only move very slowly.

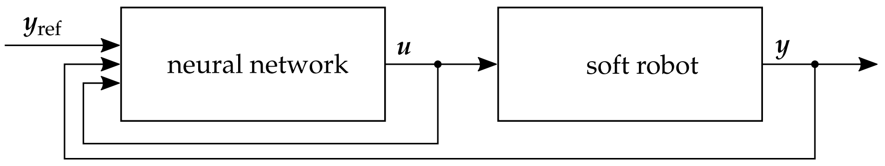

Figure 6. Block diagram of the closed control loop with the neural network as controller.

3.2. Linear Quadratic Control with Gain Scheduling

The second controller considered is a linear quadratic controller with integral action (LQI controller) and gain scheduling. This is an approach from optimal control that has not been applied to soft robotics so far. In general, for the control of linear time-invariant systems of the form

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{24}$$

LQI controllers can be used [42]. Here, $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ the input vector and $\boldsymbol{y}$ the output vector. The dynamic behavior of the linear system and the relationship between the input vector $\boldsymbol{u}$, the states $\boldsymbol{x}$ and the output $\boldsymbol{y}$ are described by the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$. As LQI controllers are extensively discussed in the literature, e.g., [43,44], they are not described in further detail here.

The controller considered so far requires a linear model of the system to be controlled for the design. However, the soft robot used in this contribution has nonlinear behavior, which can be described as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}). \end{aligned} \tag{25}$$

In order to derive an LQI controller for such a nonlinear system, it has to be linearized [45]. In this paper, the nonlinear system is therefore linearized around 2288 operation points OP, which are regularly distributed in the workspace. The operation points are chosen such that the velocities of the soft robot are zero in these points. They are shown in Figure 7. Each operation point consists of a state vector $\boldsymbol{x}_{\mathrm{OP}}$, a control variable vector $\boldsymbol{u}_{\mathrm{OP}}$ and the corresponding system output $\boldsymbol{y}_{\mathrm{OP}}$. However, as long as no external forces are considered, as in this contribution, three parameters are sufficient to describe an operation point because the steady-state configuration of the soft robot then only depends on the three tendon forces $u_1 \dots u_3$. Here, the curvature of the first, third and fifth segment, β_1, β_3 and β_5, is chosen. These can be collected in the scheduling vector $\boldsymbol{\sigma} = [\beta_1\, \beta_3\, \beta_5]^{\mathrm{T}}$.

Given the state function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, the output function $\boldsymbol{g}(\boldsymbol{x})$ and the operating point, the matrices of the state space representation of the linearized system can be determined. These are calculated, as shown by [45], among others, as

$$\widetilde{\boldsymbol{A}} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{\mathrm{OP}}, \boldsymbol{u}_{\mathrm{OP}}}, \tag{26}$$

$$\widetilde{\boldsymbol{B}} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}_{\mathrm{OP}}, \boldsymbol{u}_{\mathrm{OP}}}, \tag{27}$$

$$\widetilde{\boldsymbol{C}} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{\mathrm{OP}}, \boldsymbol{u}_{\mathrm{OP}}}. \tag{28}$$

This results in the linearized system

$$\begin{aligned} \dot{\widetilde{\boldsymbol{x}}} &= \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{u}}, \\ \widetilde{\boldsymbol{y}} &= \widetilde{\boldsymbol{C}}\widetilde{\boldsymbol{x}} \end{aligned} \tag{29}$$

with $\widetilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_{\mathrm{OP}}$, $\widetilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{y}_{\mathrm{OP}}$ and $\widetilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{OP}}$.

For each of the linearized systems in the operation points, a separate LQI controller is now designed. For the control of the soft robot, in each time step, a trilinear interpolation is performed between the eight controllers spanning the cuboid in the three-dimensional parameter space in which the current scheduling parameter $\boldsymbol{\sigma}$ lies. In Figure 8, the block diagram of the LQI controller with gain scheduling is shown. The control law results in

$$\boldsymbol{u}(\boldsymbol{\sigma}) = \boldsymbol{K}(\boldsymbol{\sigma})(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{OP}}(\boldsymbol{\sigma})) + \boldsymbol{K}_{\mathrm{i}}(\boldsymbol{\sigma}) \int (\boldsymbol{y} - \boldsymbol{y}_{\mathrm{ref}})\mathrm{d}t + \boldsymbol{u}_{\mathrm{OP}}(\boldsymbol{\sigma}). \tag{30}$$

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

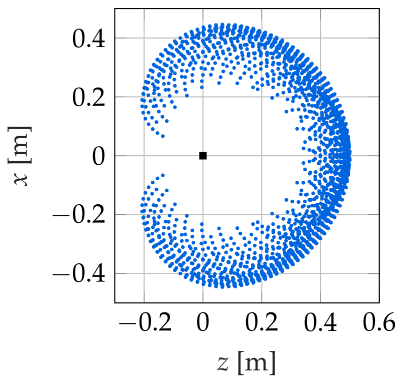

Figure 7. Tip position and spanned workspace of the LQI controller.

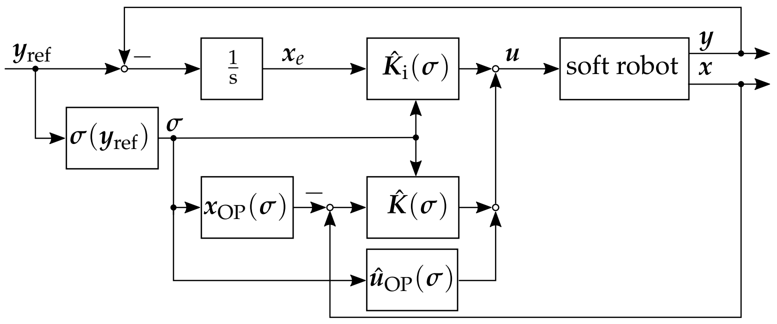
y_{ref}
−
$\frac{1}{s}$
x_e
$\hat{K}_{\mathrm{i}}(\sigma)$
u
soft robot
y
x
$\sigma(y_{\mathrm{ref}})$
σ
$x_{\mathrm{OP}}(\sigma)$
−
$\hat{K}(\sigma)$
$\hat{u}_{\mathrm{OP}}(\sigma)$

Figure 8. Block diagram of the closed control loop with the LQI controller with gain scheduling.

3.3. Model Predictive Control

The third control method considered is a model predictive controller (MPC). The block diagram of this controller is shown in Figure 9. This discrete, model-based method is, e.g., used by [25] for the control of a soft robot. The MPC solves an optimization problem in each timestep of the control in order to determine the best possible control output u. For this purpose, in every timestep, the controller optimizes the control output for the next c timesteps (control horizon) such that the control error and control effort over the next p timesteps (prediction horizon) is minimized. The optimization problem is described by

$$\min_{\boldsymbol{p}\in\mathcal{P}} J(\boldsymbol{x}(k),\boldsymbol{u}(\boldsymbol{p})) \quad \mathcal{P}=\{\boldsymbol{z}\in\mathbb{R}^{m\cdot c}\} \tag{31}$$

with the cost function

$$J(\boldsymbol{x}(k),\boldsymbol{u}) = \sum_{i=1}^{p}\Big(\boldsymbol{e}(k+i)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}(k+i)+\boldsymbol{u}(k+i)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}(k+i)+\Delta\boldsymbol{u}(k+i)^{\mathrm{T}}\boldsymbol{R}_{\Delta}\Delta\boldsymbol{u}(k+i)\Big) \tag{32}$$

where

$$\boldsymbol{e}(n) = \boldsymbol{y}_{\mathrm{ref}}(n)-\boldsymbol{y}(n), \tag{33}$$
$$\Delta\boldsymbol{u}(n) = \boldsymbol{u}(n)-\boldsymbol{u}(n-1). \tag{34}$$

Here, $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{R}_{\Delta}$ are weighting matrices to specify the influence of the different cost terms. In total, $m\cdot c$ variables have to be computed in an optimization, where m is the number of control variables. The parameters for the optimization are obtained by combining the control variables of the individual time points in the control horizon in the vector

$$\boldsymbol{p} = \begin{bmatrix}\boldsymbol{u}_{k+1}^{\mathrm{T}} & \boldsymbol{u}_{k+2}^{\mathrm{T}} & \cdots & \boldsymbol{u}_{k+c}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}. \tag{35}$$

Here, $\boldsymbol{u}_n$ is the control variable that applies to the time step with index n in the control horizon. To solve the optimization problem, in every timestep, the nonlinear soft robot model integrated into the controller has to be evaluated several times.

In this paper, a sample time of $T_\text{s} = 0.1\,\text{s}$, a prediction horizon of $p = 5$ steps and a control horizon of $c = 3$ steps are chosen. Usually, better control results can be archived with a shorter sample time and longer prediction and control horizons. However, the computational costs also increase because the optimization problem is harder to solve.

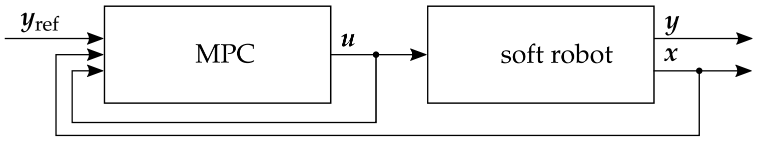

Figure 9. Block diagram of the closed control loop with the model predictive controller.

The controller is implemented in SIMULINK with the *Model Predictive Control* Toolbox. The internal model is integrated with the explicit Euler method with stepsize T_s. The optimization problem is solved with the MATLAB function *fmincon*. It uses the SQP method for the minimization [46]. With the chosen implementation of the MPC in SIMULINK, a time delay of one control step (0.1 s) is induced. However, with a different implementation of the MPC, this could be avoided.

4. Results

In the following, for all three controllers, the trajectory tracking results as well as the results of the examination of the robustness against parameter uncertainty are presented. Thereby, the control error of the tip position is calculated for an arbitrary time t as

$$e(t) = \|\boldsymbol{y}(t) - \boldsymbol{y}_\text{ref}(t)\|_2. \tag{36}$$

4.1. Test Trajectories

For the examination of the performance of the three controllers, three different types of trajectories are considered: A step change in the tip position, a swing trajectory passing through nearly the whole workspace and an L-shaped trajectory. These are shown in Figure 10. For all trajectories, only the tip position is defined; the controllers are free to choose any configuration of the soft robot that reaches that tip position. The configurations shown in Figure 10 are the configurations archived with the LQI controller. Note that the step trajectory is not continuous, and all three trajectories are not differentiable. Therefore, the controllers cannot follow the trajectories exactly.

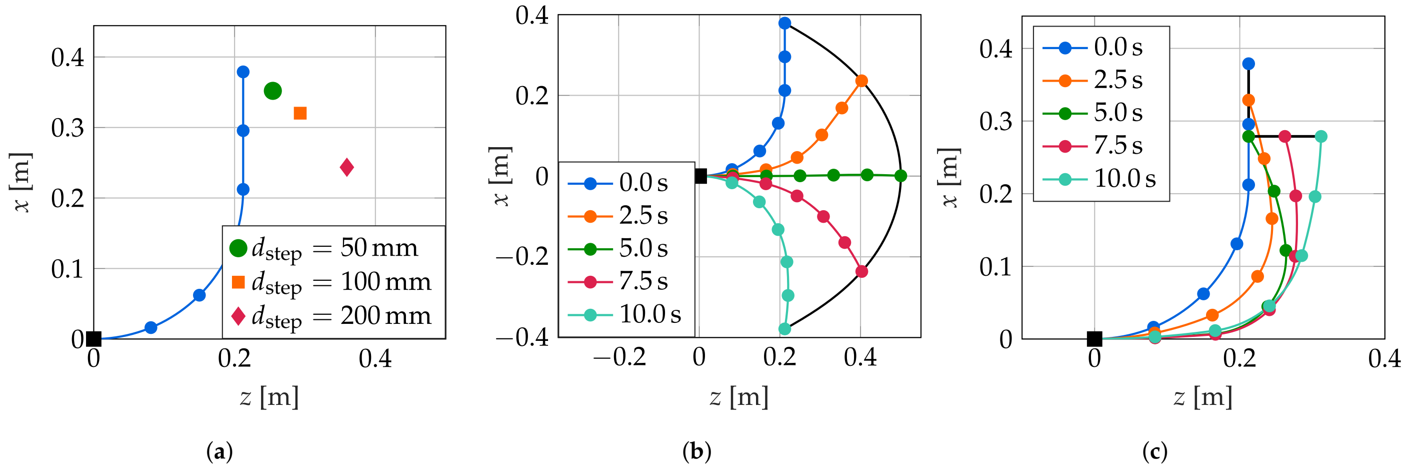

Figure 10. Examined trajectories. (**a**) Step trajectory. (**b**) Swing trajectory. (**c**) L-shape trajectory.

The step trajectory is the most basic trajectory of the three considered trajectories and is especially popular in linear control but is also widely used for nonlinear systems. Here, 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 examined. The swing trajectory crossing through nearly the whole workspace is the most natural movement of the soft robot. Starting at rest from an initial position, the soft robot has to reach its final position within 10 s with constant velocity and come to a rest there. The trajectory is not differentiable at its beginning and at its end due to the desired constant velocity. The L-shape trajectory consists of two axis-parallel movements with constant velocity that take 5 s each. Again, the soft robot has to come to a rest at the final position. This movement is less natural and can only be archived with larger curvatures.

4.2. Step Trajectory

As the first control scenario, the step change in the tip position is examined, starting with the stepsize $d_{\text{step}} = 50\,\text{mm}$. In Figure 11, the tip position is plotted over time in $x-$ and $z-$direction for all three controllers together with the desired trajectory. Characteristic quantities for the step responses are the rise time, the settling time, the overshoot and the remaining control deviation. These are listed in Table 3. Here, the rise time is the time the response takes to rise from 10% to 90% of the way from the initial value to the steady-state value. For the settling time, a band of 2% around the final rest position is considered.

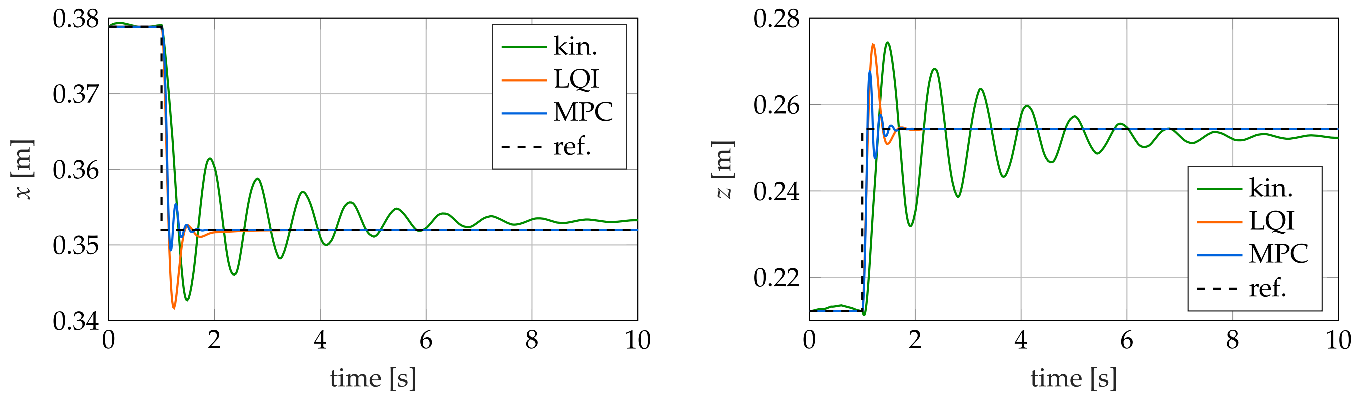

Figure 11. Reference trajectory as well as the tip position for different controllers with $d_{\text{step}} = 50\,\text{mm}$ over time.

Table 3. Characteristic quantities of the step trajectory with $d_{\text{step}} = 50\,\text{mm}$.

	Kinematic	**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%

It can be seen that all control methods considered are capable of bringing the tip of the soft robot to the desired reference position, but the quality of control partly differs clearly between the methods. It is particularly noticeable that the kinematic controller has more than twice the rise time compared to the two dynamic controllers. At the same time, the settling time of the kinematic controller is also about one order of magnitude larger. This can be explained by the fact that the kinematic controller does not consider the dynamic behavior of the soft robot. Therefore, it cannot actively damp the excited oscillations of the robot. This leads to a significantly longer settling time and reduces the maximum useful speed of the robot. The LQI and the MPC controller achieve a comparably low rise time. However, with 0.49 s, the settling time of the model predictive controller is about 45% lower than the settling time of the LQI controller. This can be explained by the predictive control behavior of the model predictive controller. This also explains the about 15% lower overshoot of the model predictive controller compared to the other two controllers considered.

Both the LQI and MPC controllers exhibit a negligible steady-state error. For the LQI controller, this can be explained by the integral control behavior. The model predictive controller achieves a negligible control error because its internal prediction model matches the soft robot model used exactly. Therefore, the necessary control variables can be determined exactly from the optimization problem. The kinematic controller has a steady-state error of 4.85%. This can be explained by the used neural network, which only represents an approximation of the inverse kinematics.

For larger steps of $d_{\text{step}} = 100\,\text{mm}$ and $d_{\text{step}} = 200\,\text{mm}$, the behavior of the controllers is qualitatively comparable. The characteristic quantities for the step responses with $d_{\text{step}} = 200\,\text{mm}$ are listed in Table 4. Only the settling time of all controllers nearly doubles or triples. This is especially noticeable for the kinematic controller since it has a very long settling time anyway. Therefore, large step sizes should be avoided when using the kinematic controller.

The calculation time of all three controllers is sufficient for real-time applications. The kinematic controller requires 5.4 µs per evaluation, which is much lower than the sample time $T_S = 200\,\text{ms}$ of the controller. Note that, as described in Section 3.1, even though the computation time is very low, the sample time should not be chosen much lower to keep the controller stable. With 11.2 µs per evaluation, the LQI controller is slightly slower. The model predictive controller takes 49.7 ms for the calculation of one control step. As expected, it has the longest computation time per evaluation of the three considered controllers. Nevertheless, the model predictive controller with a used sampling time of $T_s = 0.1\,\text{s}$ is also real-time capable on the used hardware and with the chosen prediction and control horizons. The calculation times and sample times of the controllers are listed in Table 5. For the other trajectories, comparable computation times are archived. These will therefore not be discussed further in the following.

Table 4. Characteristic quantities of the step trajectory with $d_{\text{step}} = 200\,\text{mm}$.

	Kinematic	**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%

Table 5. Computation time and sample time of the three controllers for the step trajectory.

Controller	**Sample Time**	**Calculation Time per Control Step**
kinematic	200 ms	5.4 µs
LQRI	1 ms	11.2 µs
MPC	100 ms	49.7 ms

4.3. Swing Trajectory

The time courses of the trajectories in x- and y-directions, as well as the position error e, are shown in Figure 12 for the three examined controllers. It can be seen that all the examined controllers are in principle able to follow the given trajectory. However, the control error e of the controllers differs significantly.

The largest control error of all controllers is observed for the kinematic controller. The maximum error is 27.3 mm (5.46% related to the total length of the soft robot), which is more than twice the peak error of the model predictive controller and four times that of the LQI controller. From the plots of the tip position over time, it is evident that the kinematic controller oscillates around the nominal trajectory. This can be mainly explained by the controller neglecting the dynamic characteristics of the soft robot. In addition, there are inaccuracies in the prediction of the required values for the control variables by the neural network. The kinematic controller also takes the longest time after the completion of the swing motion to reach the final position. Within the additional time of 5 s considered after the completion of the motion, it does not succeed in reaching the target position completely. Considering longer simulation times, the oscillation decays due to the internal damping of the soft robot, and a permanent control error of 1.7 mm (0.34%) remains.

The LQI controller shows the smallest control error of the examined controllers with a maximum error of 6.13 mm (1.23%) shortly after the motion starts. A second significant error occurs at time $t = 10$ s when the given swing motion ends and the tip remains in the final position. The cause of both errors is that the swing trajectory, as described in Section 4.1, is not continuously differentiable at these points. Therefore, here, the transient behavior of the system and the overshoot behavior of the controller can be observed. For the rest of the motion, the error is smaller than 1 mm (0.2%).

The model predictive controller has a maximum control error of 11.77 mm (2.35%). The errors occurring during the movement are nearly constant at that level. They vary by only about 1 mm during the trajectory. This can be seen in Figure 12 where the tip trajectory as well as the control error are plotted over time. The reason for the control error is mainly the used implementation of the MPC, which, as described in Section 3.3, induces a small time lag. The lag also shows in the tip trajectories as a nearly constant offset during the motion. This cause is also supported by the fact that the control deviation drops to nearly zero as soon as the swing movement ends and the reference signal is constant.

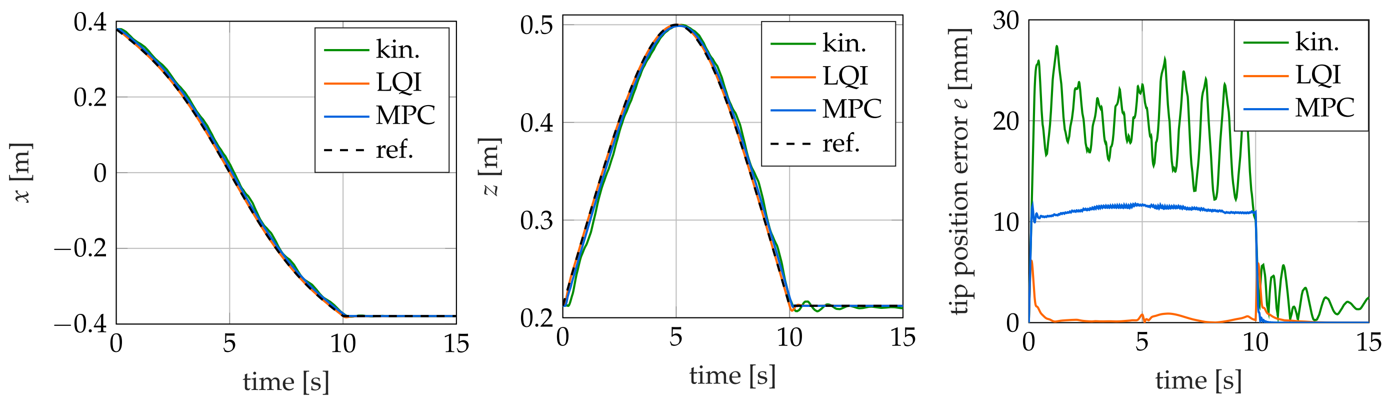

Figure 12. Reference trajectory as well as the tip position for different controllers for the swing trajectory over time.

In addition to the time courses for the tip position and the control error, the values for the control variables applied by the controllers are also of interest. These are shown for the different controllers in Figure 13 over time. Note that a positive force is an actuation of the upper tendon and a negative force is an actuation of the lower tendon. It is noticeable that all controllers calculate different values for the control variables. The basic form of the curves and the total control effort are similar, but the actual values differ by up to 29 N. Large differences are particularly noticeable for the LQI and the model predictive controller. This can be seen especially well at the end of the simulation time where both the LQRI and the MPC have a negligible control error. Furthermore, both controllers calculate different control forces. However, it follows from the different values for the control variables that the curvatures of the soft robot archived by the controllers are also different. This can be explained by the controllers finding different robot configurations that lead to the same tip position. This is also true, to a certain extent, for the kinematic controller, although it does not reach the target position completely. The configuration determined by it is also different from those of the other two controllers. The differences in the actuation forces calculated by the three controllers are only possible because the soft robot model is a kinematically redundant system. This means that there are different configurations and hence values for the control variables that lead to the same tip position.

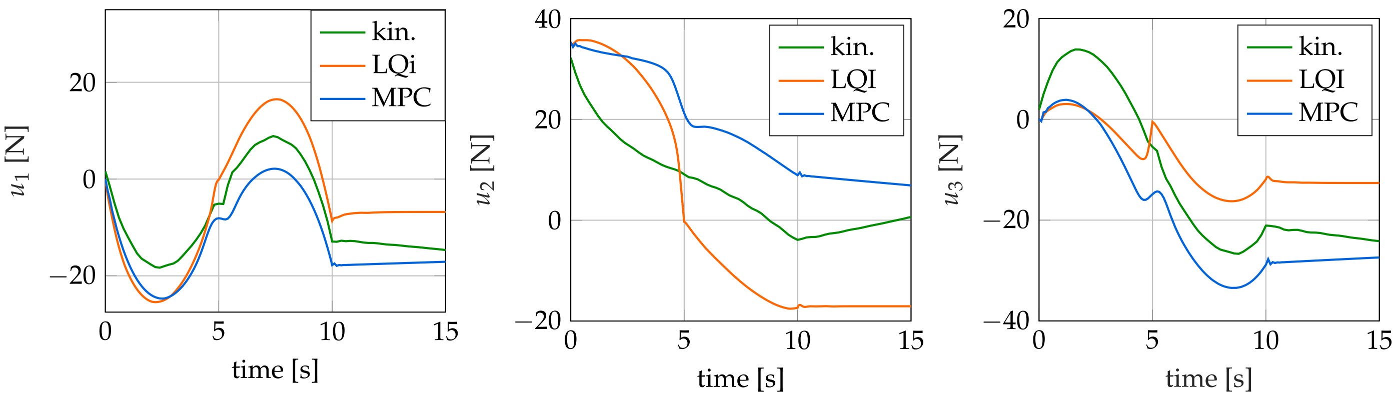

Figure 13. Actuation forces for the swing trajectory over time.

4.4. L-Shape Trajectory

The results for the control of the L trajectory are shown in Figures 14 and 15 and largely match those already observed for the swing trajectory in Section 4.3. In particular, they turn out even better for the dynamic controllers.

For the kinematic controller, the control error is again significantly larger than for the other two controllers. Thus, with a maximum value of 21.57 mm (4.31%), this is at the same level as for the swing motion. From Figure 14, it is clear that the controller has particular difficulties in reaching the final position along the x-axis. The deviation at time $t = 5$ s for this component is 20.37 mm. A possible reason for this very large deviation, also compared to the swing motion, is the considered workspace. For the design of the kinematic controller in Section 3.1, maximum actuating forces of ± 20 N are considered. The controller does not have any data from the design about the relationship between the control variables and the tip position outside this area and therefore has to extrapolate there. However, to follow the trajectory, actuation forces of $\approx \pm 40$ N are required. The control variables calculated by the neural network therefore differ from those actually required. Only at the end of the movement, when the required tip position is again in the workspace of the kinematic controller, does the occurring control error decrease again. If the time simulation is continued for a longer simulation duration, a permanent control error of 5.47 mm (1.09%) is obtained, which is significantly higher than the permanent error for the swing motion.

The maximum control error of the LQI controller with 1.71 mm (0.34%) is below the value determined for the swing motion. It is well recognizable that the position deviation always increases abruptly at the times when the trajectory is not continuously differentiable. The smaller deviation, compared to the swing motion, is due to the fact that the motion is slower because of the shorter trajectory. Thus, the steps in the velocity course of the trajectory are smaller at the points where it is not continuously differentiable. It is therefore easier for the controller to follow the trajectory and to compensate for the error that occurs. Apart from the three sudden increases and the following decay, the control error of the LQI controller is below 0.2 mm (0.4%).

With a maximum error of 3.11 mm (0.62%), the model-predictive controller also has a significantly lower control error compared to the swing trajectory. As with the latter, the deviation is at a roughly constant level during the motion. The lower occurring control error is, just as in the case of the LQI controller, due to the slower speed of the movement. As the control error is mainly caused by the lag, as explained in Section 4.3, it is nearly proportional to the velocity. This is again supported by the fact that the deviation quickly approaches zero as soon as the reference trajectory has reached the end position.

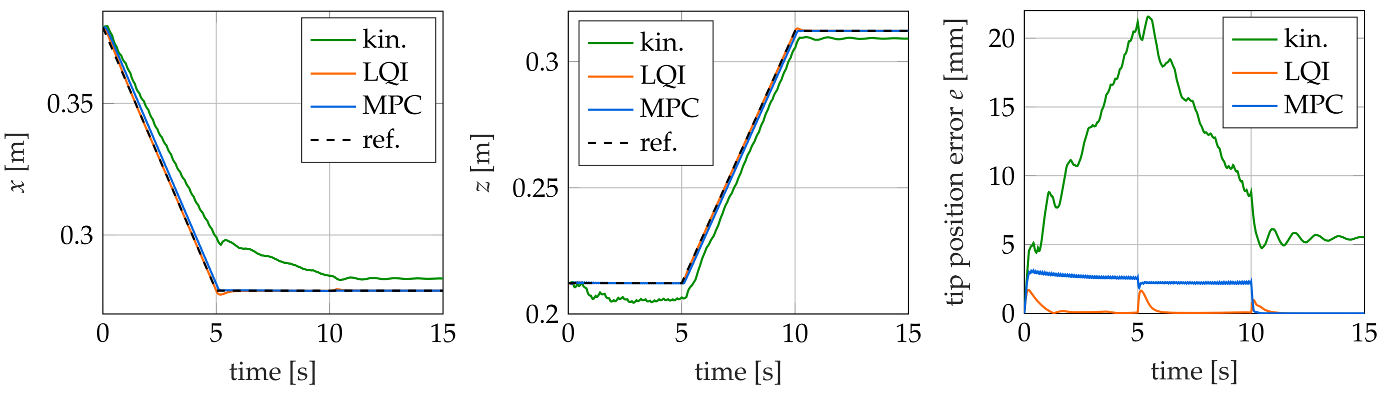

Figure 14. Reference trajectory as well as the tip position for different controllers for the L-shape trajectory over time.

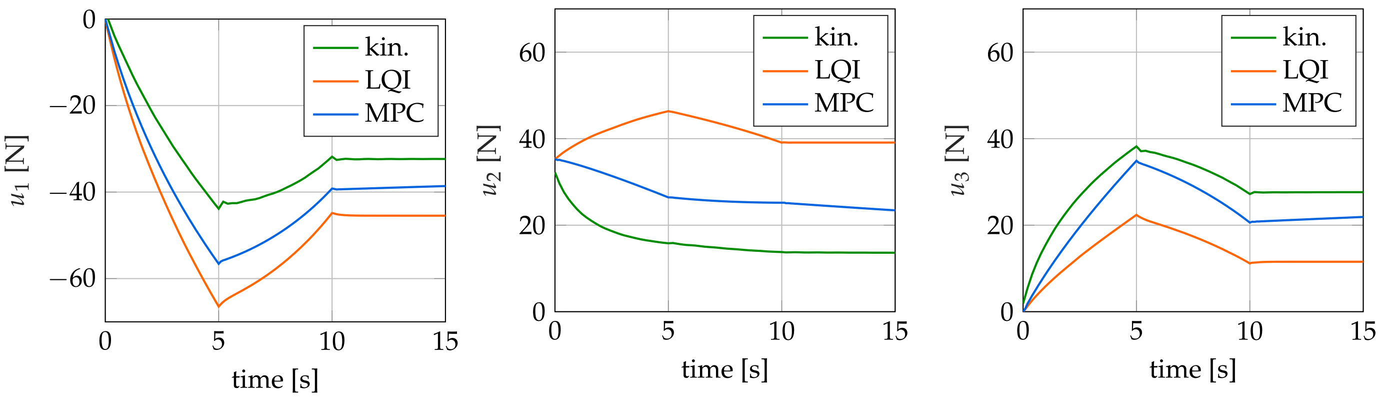

Figure 15. Actuation forces for the L-shape trajectory over time.

Finally, the control variables calculated by the controllers for the executed movement are considered. These are shown in Figure 15 for the individual controllers over the simulation time. Once again, it can be seen that the controllers determine different values for the control variables. Therefore, for this trajectory, the occurring robot configurations are also different depending on the controller. Furthermore, it is noticeable that the calculated values for the control variables are larger than in the case of the swing motion. For the kinematic controller, the maximum value of the control, 43.85 N, is more than twice as large as the 20 N considered during the design.

5. Robustness against Parameter Uncertainty

Finally, the robustness of the different control concepts against deviations in the material parameters of the robot to be controlled is investigated. Especially for soft materials, the material parameters are often not known precisely, contain unmodeled nonlinear effects, are hard to determine and might even change over time [47]. Therefore, a robustness of the controllers against parameter uncertainty is of importance.

In order to examine the influences that parameter uncertainties have on the control performance of the different controllers, deviations in the value of the Young's modulus E are considered as examples. It is often difficult to determine this value exactly, and at the same time, it has a large influence on the static and dynamic properties of the soft robot.

For the examination, for each controller, five simulations with different values of the Young's modulus in the simulation are performed. Thereby, the nominal value of the Young's modulus $E_0 = 7.32 \times 10^5$ Pa as well as a change in the Young's modulus by $\pm 25\%$ and $\pm 50\%$ are considered. In all cases, the controllers are designed for the nominal Young's modulus E_0. Note that a change in the Youngs's modulus also affects the internal damping of the soft robot model since, in this work, stiffness-proportional damping is assumed. As an example, in this work, only the tracking of the swing trajectory is considered in the robustness analysis. For the other trajectories, qualitatively comparable results could be achieved.

In Figures 16–18, the trajectory of the tip position is plotted for all three controllers and different values of the Young's modulus in the simulation model. As described in Section 5, the controllers are designed for the nominal modulus of elasticity E_0. The different controllers react very differently to this modeling error.

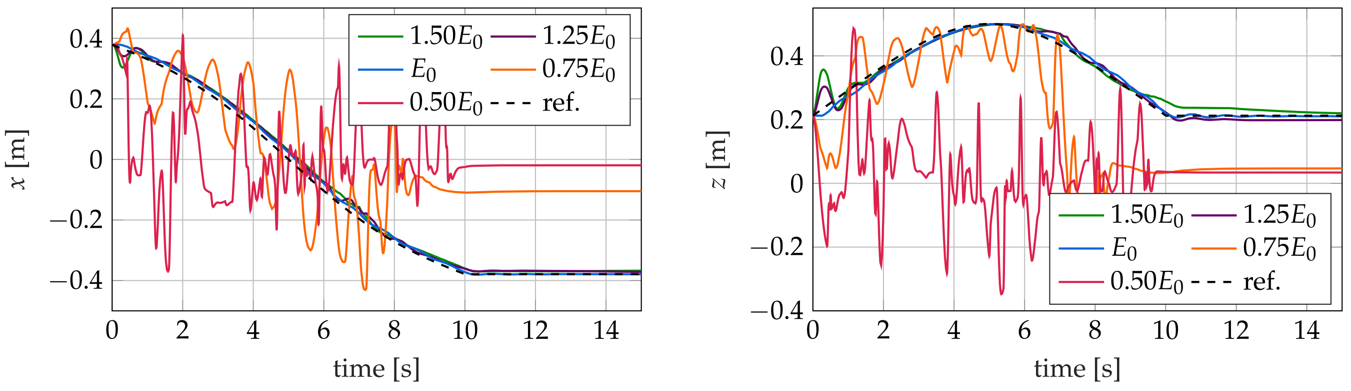

Figure 16. Trajectories of the tip position for the kinematic controller for different values of the young's modulus E.

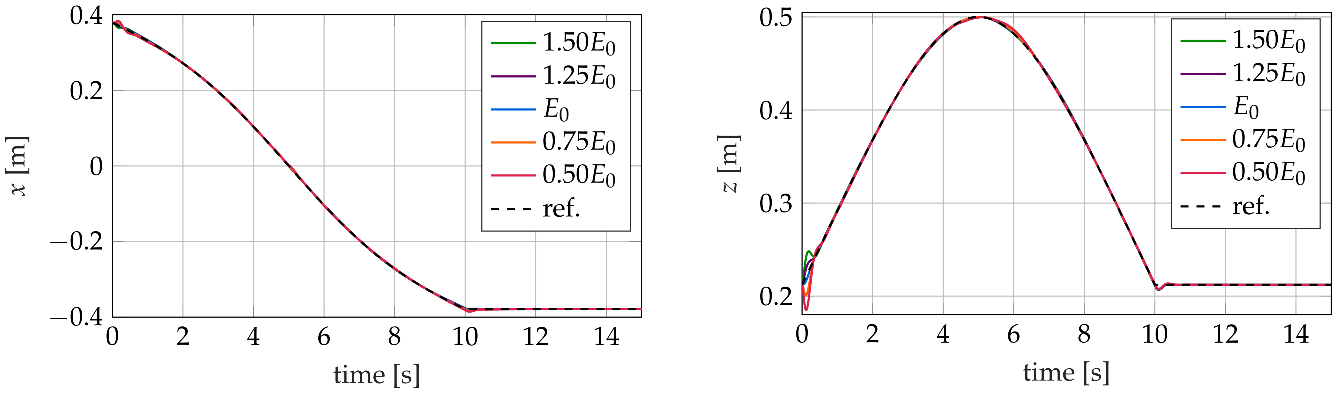

Figure 17. Trajectories of the tip position for the LQRI for different values of the young's modulus E.

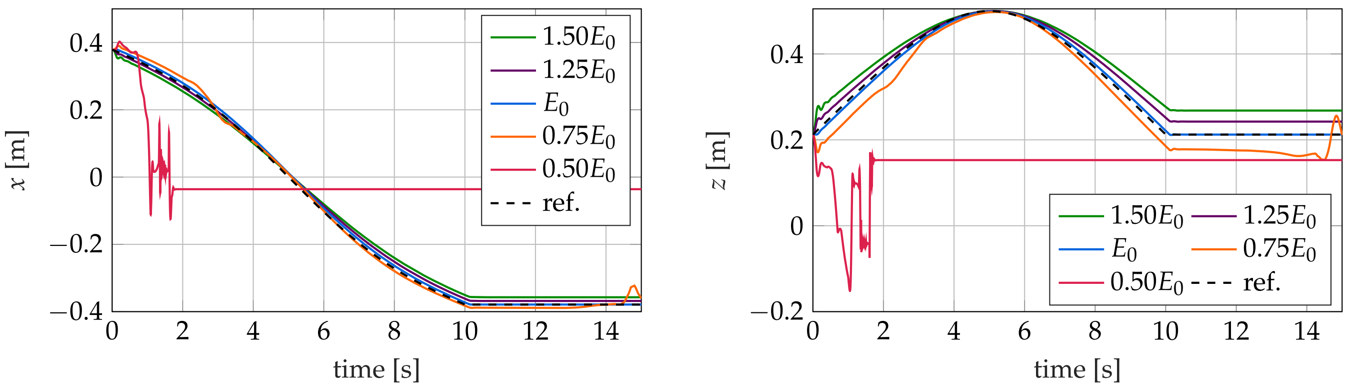

Figure 18. Trajectories of the tip position for the MPC for different values of the young's modulus E.

5.1. Kinematic Controller

The kinematic controller again shows the worst performance of the three controllers. While the control of the robot with increased Young's moduli ($E > E_0$) shows similar control results as those obtained when considering the nominal system with $E = E_0$, the determined tip position for softer robots with $E < E_0$ differs significantly. For these, an unsteady oscillatory behavior is shown before the tip comes to rest after the reference position no longer changes. However, the rest position is far away from the desired position. The reason for this is that due to the lower Young's modulus, the applied control forces result in significantly larger motions than expected by the controller. As a result, the tip of the soft robot clearly leaves the workspace considered when designing the controller. Additionally, these large motions strongly excite the dynamics of the soft robot, which only decay very slowly. This makes the control for the kinematic controller even more difficult. Thus, the controller is no longer able to determine the required control variables with sufficient accuracy. This leads to the observed unsteady oscillation behavior. This behavior is also favored by the change in the dynamic behavior of the soft robot and the decrease in its internal damping. If, on the other hand, the Young's modulus of the robot is greater than assumed by the controller, control variables that are too small are initially applied. In this case, the robot remains in its workspace, the dynamics of the controller are only slightly excited and the controller achieves a control result that is almost as good as that with the nominal Young's modulus. The higher internal damping also probably has a positive influence on the achieved control quality.

5.2. LQI Controller

In contrast to the other two controllers investigated, the LQI controller shows no noteworthy differences between the trajectory tracking with the changed Young's modulus and the nominal Young's modulus of the simulation. Note that the gains of the controller are only determined for the nominal values. There are only minor deviations at the beginning of the movement. The reason for this is the changed static behavior of the soft robot model. As a result, the controller initially determines the control variables that belong to the equilibrium position of the robot at the current operating point with a comparatively large error. This is compensated by the controller after a short time. The LQI controller has no difficulties with the changes also occurring in the dynamic behavior of the controlled robot. The reasons for this are the integral component of the controller, the large gain of the controller and the high sampling frequency of $f = 1\,\mathrm{kHz}$, which allows the controller to react quickly to changes.

5.3. MPC

Furthermore, for the model predictive controller, differences in the achieved control performance and occurring deviations can be observed depending on the Young's modulus of the controlled system. For stiffer robots with $E > E_0$, the z component of the tip position is permanently above the corresponding reference curve. The curvatures of the controlled robot turn out to be too small. As in the case of the kinematic and the LQI controller, this is due to the changed static properties of the soft robot. The control variables of the model predictive controller are determined by solving an optimization problem in such a way that they lead to the desired motion in the controller-internal model that uses the nominal parameters. If these are applied to the system to be controlled with a larger modulus of elasticity, there is less curvature than predicted by the controller due to the stiffer behavior of the robot. The too-low curvature for stiffer robots with $E > E_0$ can also be seen in the course of the x component of the tip point trajectory.

The presented effects also occur, with the same reasoning but opposite sign, for the softer robot with $E = 0.75E_0$. For this robot, the curvature is smaller than predicted by the controller. If the elastic modulus of the controlled robot is further reduced, see $E = 0.5E_0$, it becomes apparent that the MPC is no longer able to follow the reference trajectory. The low stiffness, as well as the resulting change in dynamic and static behavior, causes the closed loop to become unstable. Thus, the model predictive controller calculates ever increasing control variables to follow the reference trajectory with the internal model. These lead to the fact that the curvature of the soft robot increases more and more, and finally, the soft robot just "rolls up". From the time $t = 1.3\,\mathrm{s}$ on, the optimizer does not converge anymore and is no longer able to determine a solution for the optimization problem on which the control is based. From this moment on, the control variables of the controller are kept constant at the last "successfully" determined value. Due to the internal damping, the soft robot oscillates to the corresponding equilibrium position and remains in this position.

6. Conclusions

In this article, one kinematic and two different dynamic control approaches were analyzed for soft robots. For this purpose, these controllers were used to track various trajectories in simulation using an exemplary soft robot model. All three control methods are suitable for the control of soft robots. However, they strongly differ in implementation effort, achievable accuracy and robustness. Therefore, for these controllers, different applications arise. The advantages and disadvantages of the controllers are explained in the following and summarized in Table 6.

Table 6. Advantages and disadvantages of the examined controllers.

	Kinematic	**LQRI**	**MPC**
Online/offline implementation effort	low	medium	high
Computation effort	very low	medium	high
Accuracy	low	very high	high
Robustness	low	very high	high

6.1. Kinematic Controller

The simplest of the three controllers studied is the kinematic controller. This can be seen as the standard approach in soft robotics. This controller is model-free. The data required for training the neural network on which the controller is based can for example be determined experimentally. However, compared to the dynamic controllers, the archived control errors are large. This is especially the case for fast motions. Here, large oscillations around the desired trajectory often occur. Additionally, the examination of the robustness has shown that the stiffness of the robot must not be overestimated by the controller to allow trajectory tracking. Additionally, this controller requires very low computational effort. Therefore, the kinematic controller should be used if low online implementation effort is important and/or no accurate model of the soft robot is available, while the required movements of the robot are slow and medium control errors, as well as oscillations, can be accepted. This holds for most of the current soft robotic applications.

6.2. LQRI

In this paper, an LQI controller with gain scheduling was proposed, which has not been used before in soft robots' literature. This is the most accurate and robust of the three examined controllers. At the differentiable parts of the trajectory, the control error is negligible. Small control errors can only be observed at the non-differentiable parts of the trajectory. The LQRI is very robust against parameter uncertainties. Even a large change in the Young's modulus of $\pm 50\%$ leads to small control errors. The controller has moderate computational effort (11.2 µs per iteration on a standard PC). The LQRI is a good option if the performance of the kinematic controller is not good enough. It can archive much higher accuracy and robustness than the kinematic controller, resulting in much higher achievable speeds. However, the implementation and computational effort is higher.

6.3. MPC

The model predictive controller achieves a comparatively high accuracy as the LQRI. However, due to the implementation used in this contribution, a small time delay cannot be avoided. The controller is model-based and therefore requires an accurate model of the robot. The MPC is robust against moderate parameter uncertainties. However, the control error increases significantly with increasing difference between the controlled robot and the internal model of the robot. If the stiffness of the robot is strongly underestimated by the MPC, it fails to find a solution for the optimal control problem. The computational cost of the MPC is very high compared to the other two examined controllers, but depending on the choice of control and prediction horizon, it is real-time capable on a standard PC. The MPC is a good alternative to the LQRI, especially if an accurate model of the robot is available and high computational costs are not a problem. As an advantage, additional constraints, such as actuator limits and avoiding obstacles, can easily be included in the optimization problem.

6.4. Limitations and Perspectives

The main limitation of this work is that the results were only obtained in simulations so far. In the future, it is planned to experimentally investigate what extent the results can be transferred to applications with physical robots. Additionally, the controllers can be extended to the control of movements in 3D.

Author Contributions: Conceptualization, M.G. and R.S.; methodology, J.C.W., M.G. and R.S.; software, J.C.W.; validation, J.C.W. and M.G.; investigation, J.C.W.; data curation, J.C.W. and M.G.; writing—original draft preparation, M.G.; writing—review and editing, M.G. and R.S.; visualization, J.C.W. and M.G.; supervision, R.S.; project administration, R.S.; funding acquisition, R.S. All authors have read and agreed to the published version of the manuscript.

Funding: Publishing fees funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Projektnummer 491268466—and the Hamburg University of Technology (TUHH) in the funding program *Open Access Publishing*.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

LQI	Linear quadratic (regulator) with integral action
LQR	Linear quadratic regulator
LQRI	Linear quadratic regulator with integral action
LWPR	locally weighted projection regression
MPC	Model predictive controller
PCC	Piecewise constant curvature
PD	Proportional derivative [controller]

References

1. Rus, D.; Tolley, M.T. Design, fabrication and control of soft robots. *Nature* **2015**, *521*, 467–475. [CrossRef] [PubMed]
2. Lee, C.; Kim, M.; Kim, Y.J.; Hong, N.; Ryu, S.; Kim, H.J.; Kim, S. Soft Robot Review. *IJCAS* **2017**, *15*, 3–15. [CrossRef]
3. Webster, R.J.; Jones, B.A. Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review. *Int. J. Robot. Res.* **2010**, *29*, 1661–1683. [CrossRef]
4. 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 inControl, Automation and Robotics, Lisbon, Portugal, 29–31 July 2016; SciTePress—Science and and Technology Publications: Setubal, Portugal, 2016; pp. 303–310. [CrossRef]
5. 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 Robot.* **2017**, *4*, 285–296. [CrossRef] [PubMed]
6. George Thuruthel, T.; Ansari, Y.; Falotico, E.; Laschi, C. Control Strategies for Soft Robotic Manipulators: A Survey. *Soft Robot.* **2018**, *5*, 149–163. [CrossRef] [PubMed]
7. Camarillo, D.B.; Carlson, C.R.; Salisbury, J.K. Configuration Tracking for Continuum Manipulators With Coupled Tendon Drive. *IEEE Trans. Robot.* **2009**, *25*, 798–808. [CrossRef]
8. 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, Barcelona, Spain, 18–22 April 2005; pp. 924–929. [CrossRef]
9. Jones, B.A.; Walker, I.D. Kinematics for multisection continuum robots. *IEEE Trans. Robot.* **2006**, *22*, 43–55. [CrossRef]

10. Renda, F.; Armanini, C.; Mathew, A.; Boyer, F. Geometrically-Exact Inverse Kinematic Control of Soft Manipulators With General Threadlike Actuators' Routing. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7311–7318. [CrossRef]
11. Camarillo, D.B.; Carlson, C.R.; Salisbury, J.K. Task-Space Control of Continuum Manipulators with Coupled Tendon Drive. In *Experimental Robotics*; Springer Tracts in Advanced Robotics; Siciliano, B., Khatib, O., Groen, F., Kumar, V., Pappas, G.J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 54, pp. 271–280. [CrossRef]
12. Marchese, A.D.; Komorowski, K.; Onal, C.D.; Rus, D. Design and control of a soft and continuously deformable 2D robotic manipulation system. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014. [CrossRef]
13. 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 Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 5033–5039. [CrossRef]
14. Rolf, M.; Steil, J. Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 1147–1160. [CrossRef]
15. Thuruthel, T.G.; Falotico, E.; Cianchetti, M.; Laschi, C. *Learning Global Inverse Kinematics Solutions for a Continuum Robot*; Springer: Cham, Switzerland, 2016; pp. 47–54. [CrossRef]
16. 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 Robot. Autom. Lett.* **2022**, *7*, 8315–8322. [CrossRef]
17. Yip, M.C.; Camarillo, D.B. Model-Less Feedback Control of Continuum Manipulators in Constrained Environments. *IEEE Trans. Robot.* **2014**, *30*, 880–889. [CrossRef]
18. Yip, M.C.; Camarillo, D.B. Model-Less Hybrid Position/Force Control: A Minimalist Approach for Continuum Manipulators in Unknown, Constrained Environments. *IEEE Robot. Autom. Lett.* **2016**, *1*, 844–851. [CrossRef]
19. Kapadia, A.D.; Walker, I.D.; Dawson, D.M.; Tatlicioglu, E. A model-based sliding mode controller for extensible continuum robots. In *Recent Advances in Signal Processing, Robotics and Automation*; Mathematics and Computers in Science and Engineering; WSEAS Press: Cambridge, UK, 2010.
20. Kapadia, A.; Walker, I.D. Task-space control of extensible continuum manipulators. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 1087–1092. [CrossRef]
21. 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. *Int. J. Robot. Res.* **2020**, *39*, 490–513. [CrossRef]
22. Falkenhahn, V.; Hildebrandt, A.; Neumann, R.; Sawodny, O. Model-based feedforward position control of constant curvature continuum robots using feedback linearization. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 762–767. [CrossRef]
23. Falkenhahn, V.; Hildebrandt, A.; Neumann, R.; Sawodny, O. Dynamic Control of the Bionic Handling Assistant. *IEEE/ASME Trans. Mechatron.* **2017**, *22*, 6–17. [CrossRef]
24. 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 Robot. Autom. Mag.* **2016**, *23*, 75–84. [CrossRef]
25. 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 Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 613–618. [CrossRef]
26. Trumic, M.; Della Santina, C.; Jovanovic, K.; Fagiolini, A. Adaptive Control of Soft Robots Based on an Enhanced 3D Augmented Rigid Robot Matching. In Proceedings of the 2021 American Control Conference (ACC), New Orleans, LA, USA, 25–28 May 2021. [CrossRef]
27. Franco, E. Model-Based Eversion Control of Soft Growing Robots with Pneumatic Actuation. *IEEE Control. Syst. Lett.* **2022**, *6*, 2689–2694. [CrossRef]
28. Engel, Y.; Szabo, P.; Volkinshtein, D. Learning to control an octopus arm with gaussian process temporal difference methods. In Proceedings of the 29th Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–10 December 2005.
29. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 387–395.
30. Zhang, H.; Cao, R.; Zilberstein, S.; Wu, F.; Chen, X. Toward Effective Soft Robot Control via Reinforcement Learning. In Proceedings of the International Conference on Intelligent Robotics and Applications, Wuhan, China, 16–18 August 2017; Springer: Cham, Switzerland, 2017; pp. 173–184. [CrossRef]
31. Thuruthel, T.G.; Falotico, E.; Renda, F.; Laschi, C. Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators. *IEEE Trans. Robot.* **2019**, *35*, 124–134. [CrossRef]
32. Hyatt, P.; Killpack, M.D. Real-Time Nonlinear Model Predictive Control of Robots Using a Graphics Processing Unit. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1468–1475. [CrossRef]
33. Bruder, D.; Fu, X.; Gillespie, R.B.; Remy, C.D.; Vasudevan, R. Data-Driven Control of Soft Robots Using Koopman Operator Theory. *IEEE Trans. Robot.* **2021**, *37*, 948–961. [CrossRef]

34. 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 Proceedings of the 2018 IEEE International Conference on Soft Robotics (RoboSoft), Livorno, Italy, 24–28 April 2018. [CrossRef]
35. Dürichen, R.; Pimentel, M.A.F.; Clifton, L.; Schweikard, A.; Clifton, D.A. Multitask Gaussian processes for multivariate physiological time-series analysis. *IEEE Trans.-Bio-Med. Eng.* **2015**, *62*, 314–322. [CrossRef]
36. Kramer, R.K.; Majidi, C.; Sahai, R.; Wood, R.J. Soft Curvature Sensors for Joint Angle Proprioception. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 1919–1926. [CrossRef]
37. Ozel, S.; Keskin, N.A.; Khea, D.; Onal, C.D. A Precise Embedded Curvature Sensor Module for Soft-bodied Robots. *Sens. Actuators A Phys.* **2015**, *236*, 349–356. [CrossRef]
38. 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 Technol. Lett.* **2016**, *28*, 2237–2240. [CrossRef]
39. Grube, M.; Seifried, R. An Optical Curvature Sensor for Soft Robots. In *ROMANSY 24—Robot Design, Dynamics and Control*; Kecskeméthy, A., Parenti-Castelli, V., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 125–132.
40. Rone, W.S.; Ben-Tzvi, P. Continuum Robot Dynamics Utilizing the Principle of Virtual Power. *IEEE Trans. Robot.* **2014**, *30*, 275–287. [CrossRef]
41. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer International Publishing: Cham, Switzerland, 2016. [CrossRef]
42. Young, P.C.; Willems, J.C. An Approach to the Linear Multivariable Servomechanism Problem. *Int. J. Control.* **1972**, *15*, 961–979. [CrossRef]
43. Hendricks, E.; Jannerup, O.; Haase Sørensen, P. *Linear Systems Control: Deterministic and Stochastic Methods*; Springer: Berlin/Heidelberg, Germany, 2008. [CrossRef]
44. Porter, B. Optimal Control of Multivariable Linear Systems incorporating Integral Feedback. *Electron. Lett.* **1971**, *7*, 170. [CrossRef]
45. Adamy, J. *Nichtlineare Systeme und Regelungen*; Springer: Berlin/Heidelberg, Germany, 2014. [CrossRef]
46. Nocedal, J.; Wright, S.J. *Numerical Optimization*, 2nd ed.; Springer series in operation research and financial engineering; Springer: New York, NY, USA, 2006. [CrossRef]
47. 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 J. Autom. Sin.* **2020**, *7*, 451–460. [CrossRef]