Dynamic Tracking Control for Soft Robots : Data-Driven Modeling and Robust Control Design

Shijie Li

Thèse de doctorat

Pour obtenir le grade de Docteur de

l'UNIVERSITE POLYTECHNIQUE HAUTS-DE-FRANCE

et de l'INSA HAUTS-DE-FRANCE

Commande de suivi dynamique pour les robots flexibles: Modélisation basée sur les données et conception de commandes robustes

Spécialité: Automatique, Productique

Présentée et soutenue par Shijie LI.
Le 27/01/2023, à Valenciennes

Ecole doctorale :

Ecole Doctorale Polytechnique Hauts-de-France (ED PHF n°635)

Unité de recherche :

Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH - UMR CNRS 8201)

JURY

Président du jury :

Christine Chevallereau	Directeur de Recherche, CNRS LS2N Nantes

Rapporteurs :

Jamal Daafouz	Professeur, Université de Lorraine
Edouard Laroche	Professeur, Université de Strasbourg

Examinatrice :

Christine Chevallereau	Directeur de Recherche, CNRS LS2N Nantes

Co-directeur de thèse :

Thierry-Marie Guerra	Professeur, Université Polytechnique Hauts-de-France
Alexandre Kruszewski	Professeur, Centrale Lille Institut

Co-encadrant :

Anh-tu Nguyen	Maître de conférences, INSA Hauts-de-France

Membres invités :

Christian Duriez	Directeur de recherche INRIA Lille

PhD Thesis

Submitted for the degree of Doctor of Philosophy from

l'UNIVERSITE POLYTECHNIQUE HAUTS-DE-FRANCE

and l'INSA HAUTS-DE-FRANCE

Dynamic Tracking Control for Soft Robots: Data-Driven Modeling and Robust Control Design

Subject: Automation, Production

Presented and defended by Shijie LI
On 27 January 2023, Valenciennes

Doctoral school:

Doctoral School Polytechnique Hauts-de-France (ED PHF n°635)

Research unit:

Laboratory of Industrial and Human Automation control Mechanical engineering and Computer science (LAMIH – UMR CNRS 8201)

JURY

President of jury :

Christine Chevallereau	Directeur de Recherche, CNRS LS2N Nantes

Reviewers :

Jamal Daafouz	Professeur, Université de Lorraine
Edouard Laroche	Professeur, Université de Strasbourg

Examiner :

Christine Chevallereau	Directeur de Recherche, CNRS LS2N Nantes

Thesis co-director :

Thierry-Marie Guerra	Professeur, Université Polytechnique Hauts-de-France
Alexandre Kruszewski	Professeur, Centrale Lille Institut

Co-supervisor :

Anh-tu Nguyen	Maître de conférences, INSA Hauts-de-France

Invited members :

Christian Duriez	Directeur de recherche INRIA Lille

Acknowledgements

First of all, I would like to thank Prof. Jamal Daafouz and Prof. Edouard Laroche for being the reviewers of my thesis and Prof. Christine Chevallereau for being the president of jury. In addition, I would like to thank Christian Duriez and defrost team for coming to my defense and for your generous assistance during my Ph.D. .

My doctoral life has gone by in a flash, while I still have the impression of that sunny May afternoon when I first came to Valenciennes for the interview. These years have left me with so many wonderful memories and a lifetime of happiness to look back on. Many thanks to Anh-Tu for selecting me for this Ph.D. project and for starting this wonderful experience for me.

Most importantly, I would like to thank my three outstanding supervisors Anh-Tu, TMG and Alex for giving me the opportunity to work on this challenging and interesting project. I am well aware that not all Ph.D. students have access to such nice research topic. Besides, they have also shown me and taught me how to conduct high quality academic research, not to mention the endless help they have given me throughout the years.

In addition, I would like to thank my colleagues and friends for working together to solve problems in our lives and research. From building platforms to discussing algorithms, from hiking together to drinking and talking, it all makes me feel happier in the Ph.D. life.

Thanks to my girlfriend Wanxin, we have known each other since high school until today and have been partners in our journey of life. Especially, it would be hard to live and study during the epidemic without your support. Many thanks to my parents and family for their consistent support, without you I could not have reached current height step by step.

About France, during these years, I drove with Wanxin and our cat, euler, in every region of France. Three years and 70,000 kilometers, we have seen countless sights, history, and lovely people. I have also developed a strong affection for this land. It is very unfortunate that my post-doc could not continue in France due to political reasons. I don't feel resentful, instead I am surprised that my life have to

take another dramatic turn.

I would like to thank again Anh-Tu who has been guiding me and working closely with me, our relationship between director and friends has touched me a lot and influenced me deeply. I hope that you will recover quickly so that you can unleash your talents in your future research.

Finally, thank you all, I love you all.

Résumé

Cette thèse porte sur les principaux problèmes de contrôle en robotique douce. Une approche étape par étape est proposée, de la modélisation du robot au contrôle dynamique et à la planification du mouvement. En plus des travaux théoriques, de nombreuses expériences et simulations sont également menées pour illustrer l'efficacité des méthodes proposées.

La thèse prolonge d'abord les travaux antérieurs de notre équipe, dans lesquels le cadre de modélisation et de contrôle linéaire est proposé sur la base du modèle de la méthode des éléments finis d'ordre réduit. Nous avons discuté en détail de l'utilisation et des avantages d'une méthode améliorée de réduction de l'ordre du modèle qui préserve la structure et la stabilité du modèle FEM. Cette nouvelle méthode de modélisation permet la conception d'un contrôleur de rétroaction-anticipation basé sur un observateur de perturbations. Cependant, le schéma de contrôle proposé est efficace mais limité à une petite plage où le modèle linéaire est efficace.

Pour effectuer le contrôle dynamique dans l'ensemble de l'espace de travail non linéaire du robot logiciel, nous avons étendu le schéma de contrôle linéaire à un nouveau cadre de contrôle à paramètres linéaires variables (LPV). Le modèle LPV est développé en utilisant à la fois des modèles d'ordre réduit obtenus à partir de modèles FEM et les données collectées à partir des robots mous. La stabilité du contrôleur généralisé est garantie par la théorie de la stabilité de Lyapunov et la conception du contrôleur est formulée comme un problème d'optimisation sous inégalités matricielles linéaires (LMI).

Cependant, le modèle LPV est basé sur des modèles linéarisés. Les tâches de commande prédictive et de planification de mouvement ne peuvent pas être réalisées efficacement avec les modèles LPV. Pour atteindre des performances plus élevées pour le contrôle des robots mous, nous avons également proposé des solutions correspondantes à ces problèmes. Une nouvelle représentation cinématique des configurations de robots mous combinant à la fois le modèle et les données collectées est proposée, ainsi qu'une méthode de cinématique inverse directe basée sur les données

de mesure. Pour la commande anticipative de robots souples qui ne peut être obtenue analytiquement, des résultats préliminaires sur la représentation fonctionnelle de la commande anticipative et des lois d'apprentissage anticipatif avec des garanties de stabilité sont illustrés.

Mots-clés: Robots souples, commande LPV, commande de mouvement, modélisation pilotée par les données, réduction de l'ordre des modèles, méthode des éléments finis, cinématique inverse, commande basée sur l'apprentissage.

Abstract

This thesis focuses on the key control problems in soft robotics. A step-by-step approach is proposed, from robot modeling to dynamic control and motion planning. In addition to the theoretical works, extensive experiments and simulations are also conducted to illustrate the effectiveness of proposed methods.

The thesis at first extends the previous work of our team, in which the linear modeling and control framework is proposed based on the reduced order Finite Element Method model. We thoroughly discussed the usage and advantages of an improved model order reduction method that preserves the structure and stability of FEM model. This new modeling method enables the design of a disturbance observer based feedback-feedforward controller. However, proposed control scheme is effective but limited to a small range where the linear model is effective.

To perform the dynamic control in the whole nonlinear workspace of soft robot, we extended the linear control scheme to a new linear parameter varying (LPV) control framework. The LPV model is developed using both reduced order models obtained from FEM models and the data collected from the soft robots. The stability of generalized controller is guaranteed with Lyapunov stability theory and the controller design is formulated as an optimization problem under linear matrix inequalities(LMI).

However, LPV model is based on linearized models. The feedforward control and motion planning tasks cannot be achieved effectively with LPV models. To achieve higher performance for the control of soft robots, we also proposed corresponding solutions for these issues. A novel kinematic representation of soft robot configurations combining both model and collected data is proposed, as well as a direct inverse kinematics method based on measurement data. For the feedforward control of soft robot that cannot be obtained analytically, preliminary results about functional representation of feed-forward control and feed-forward learning laws with stability guarantees are illustrated.

Keywords: Soft robots, LPV control, motion control, data-driven modeling, model order reduction, finite element method, inverse kinematics, learning based control.

Nomenclature

Abbreviations

Abbreviation	Definition
FEM	Finite element method
MOR	Model order reduction
POD	Proper orthogonal decomposition
EID	Equivalent Input Disturbance
FK	Forward kinematics
IK	Inverse kinematics
DoF	Degree of freedom
PCC	Piece-wise constant curvature
LPV	Linear parameter varying
LMI	Linear matrix inequality
SVD	Singular value decomposition
GPR	Gaussian Process Regressor
i.i.d.	Independent and Identically Distributed
RBF	Radial Basis Function
ILC	Iterative Learning Control
MPC	Model Predictive Control

Symbols

Symbol	Definition
$\mathbb{R}$	set of real numbers
$\mathbb{N}$	set of non-negative integers
$X^\top$	Transpose of matrix X
X^{-1}	Inverse of symmetric matrix X
$X^\dagger$	Pseudo-inverse of matrix X
$X \succ 0$	X is symmetric positive definite
$X_{(i)}$	i-th column of matrix X
$\mathrm{He}(X)$	$X + X^\top$
$\mathrm{Sym}(X)$	$X - X^\top$
$\lambda_{\min}(X)$, $\lambda_{\max}(X)$	min. and max. eigenvalues of symmetric matrix X
$\mathrm{diag}(X_1, X_2)$	block-diagonal matrix composed of matrices X_1 and X_2
I	identity matrix of appropriate dimension
$\|x\| = \sqrt{x^\top x}$	2-norm of vector $x \in \mathbb{R}^n$
$\|f\|_\infty = \sup_{t\in\mathbb{R}} \|f(t)\|$	$\ell_\infty-$norm of function $f : \mathbb{R} \to \mathbb{R}^n$
$\mathscr{B}_\infty$	set of bounded functions f
$(\star)$	Transpose quantity in an expression or in a symmetric matrix
$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$	The variable $\boldsymbol{x}$ has a Gaussian (Normal) distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ
$p(y\|x)$	conditional random variable y given x and its probability (density)

Contents

Chapter 1

Introduction

Soft robotics is the new hot spot in the field of robotics and also the natural extension of the mature rigid-body robot research. Its maturity is inseparable from the development of various fields such as computing, manufacturing, and materials, etc. The special complexity of soft robots also provides additional challenges for each direction. Among these, control is undoubtedly central to the successful application and function of soft robots. While the research about corresponding dynamic modeling and control methods is currently insufficient.

In this thesis, we conduct research on key control problems in soft robotics. A step-by-step approach is proposed, from modeling to control, and planning of the soft robot motion. As a first step we present the background of soft robotics, our research motivations and the challenges we face.

1.1 Background

Inspired by various creatures in nature, soft robots have capabilities in terms of large-scale flexibility, dexterity, compliance and adaptability [1]. These robots have been designed to perform complex tasks in interaction with humans, which involve a high degree of uncertainty, e.g., surgery, assistive medical devices, search and rescue. Up to now, considerable efforts have been devoted to the research and development [2], [3]. Various hardware solutions have been developed to push the boundaries of robot abilities [4]. Many researchers have sought to reproduce soft-bodied creatures living in nature. For example, based on caterpillar locomotion, soft inchworm robotic platforms have been designed such that they can crawl, inch or roll [5]–[7]. Flexible fish-like robots capable of swimming underwater were made using silicone rubbers. [8]–[10].

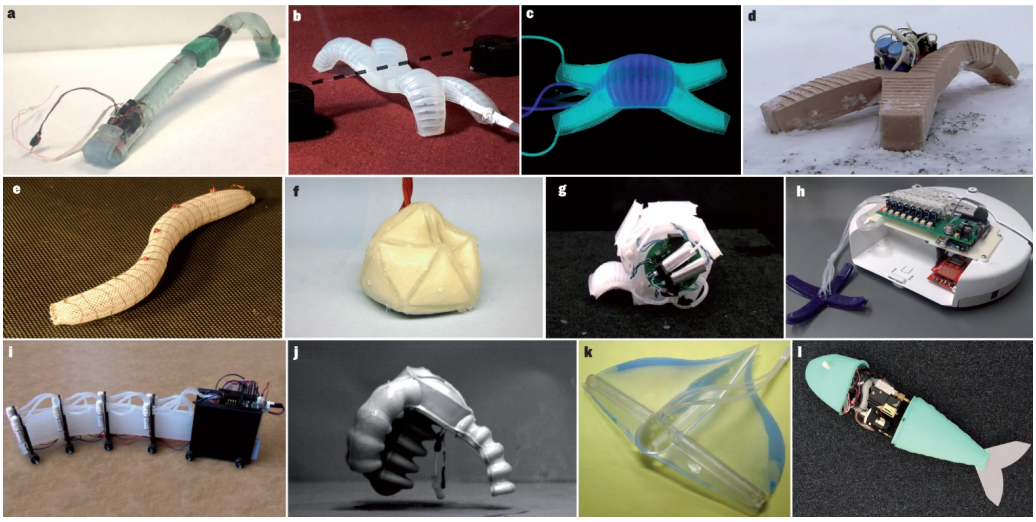


Figure 1.1: Various soft robots and manipulators [1]. a - Caterpillar-inspired locomotion. b - A multi-gait quadruped. c - Active camouflage. d - Walking in hazardous environments. e - Worm-inspired locomotion. f - Particle-jamming-based actuation. g - Rolling powered by a pneumatic battery. h - A hybrid hard–soft robot. i - Snake-inspired locomotion. j - Jumping powered by internal combustion. k - Manta-ray inspired locomotion. l - An autonomous fish.

Apart from these replication of real creatures, soft manipulators also received particular attention with the aim to achieve smooth and flexible deformations such as elephant trunks [11], [12], octopus tentacles [13], [14], or skeletal spines of verte-

brate animals [15]. Soft manipulators can be designed and fabricated with various structures, materials, and actuation technologies, like the continuum robots actuated by steel cables [16], by shape-memory alloys [17], by pneumatic artificial muscles [18], or silicone soft robots [12], [19], etc.

The design of the soft robot determines the upper limit of the performance it can achieve, and the control algorithm determines how much of the robot's performance can be utilized. Many attempts have already been made to control soft robots. Most of these efforts are either based on heuristic algorithms or are only concerned with a specific type of robot, and thus they are not able to propose a framework for control that could be clear and possibly generic. Based on the work of interdisciplinary team and former colleagues, we proposed a preliminary framework for dynamic modeling to control of soft robots. To begin with, we will first discuss the structure and mechanism of elephant trunk to illustrate the ideal soft robot and the type of control performance we require.

1.2 Elephant trunk - ideal soft robot and source of inspiration

For the survival of an elephant in the wild, its trunk is an indispensable tool. Whether it is for drinking, eating and generally interacting with the outside world, their trunk is a crucial interface. The elephant's trunk has a high degree of flexibility, moreover, flexibility without losing its strength.

An elephant trunk has about 40,000 muscle fibers divided into 16 distinct groups according to the nervous system that controls them [20]. Since the muscles can only contract, the 16 groups can be further simplified as 8 antagonistic pairs that are performing independent actions. For comparison, the human tongue is made up of 8 groups and 4 pairs of muscles. The elephant's trunk can be divided into 3 sections, each with well-defined functions as shown in Table 1.1. For more detailed information about anatomy and mechanism of an elephant's trunk, readers can refer to [20]–[22].

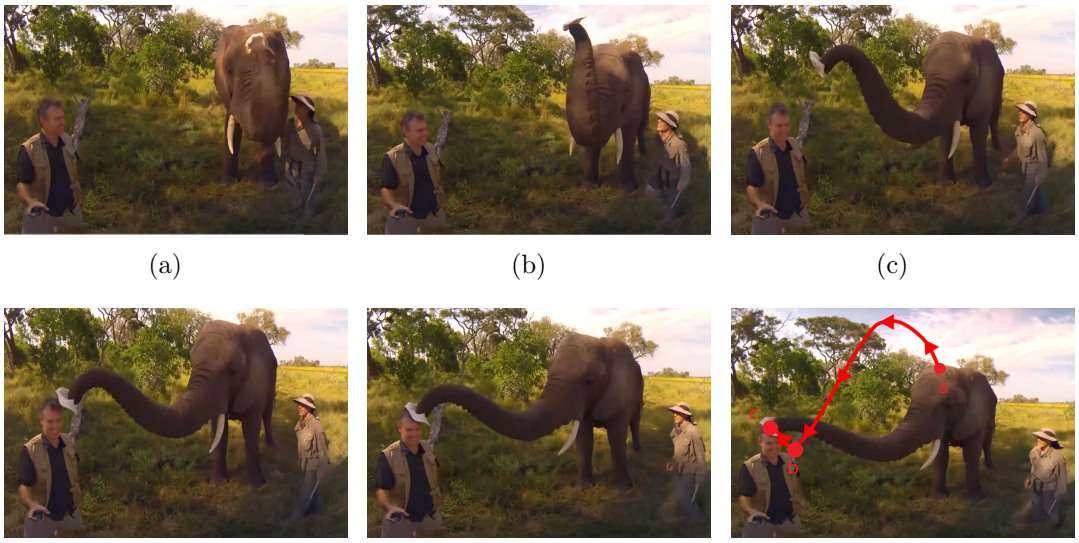
Part of trunk	Function
Section I-base	bend
Section II-middle	stretch and bend
Section III-tip	stretch, bend, twist

Table 1.1: The function of each segment of the elephant trunk.

(a) (b) (c)

Figure 1.2: Three type of elephant trunk functions (a) Bending (b) Stretching (c) Twisting.

Researchers have obtained rich results on building soft robots imitating the trunk structure with various sizes, material and actuators [23]–[25]. Whereas, several of them have surpassed the real elephant trunk in terms of strength and accuracy, their performances are still far from those of a real trunk, the limitations are due to the low detection and control performances. In relation to the research goals of this work, let us analyze the characteristics of trunk control based on the typical behavior of elephants.



(a) (b) (c)

(d) (e) (f)

Figure 1.3: The elephant puts the hat on a man's head from its own head [26]. (a)-(e) are stepwise figures and the trajectory is shown in (f).

To illustrate a typical trunk movement, Figure 1.3 shows the trajectory used to pass a hat from an elephant's head to a human's. The trajectory of the hat Figure 1.3-(f) shows that the elephant is not moving its end-effector along the shortest path (a straight line from a to c). Instead, the elephant estimates an appropriate configuration of the trunk that can place the hat on the head of the man and moves according to it. The method and criterion to generate the so-called suitable posture are also the research focuses of neuroscientists [27]. However, this first move is not fully successful, as the final position is point b (see figure 1-3f) instead of point c which is the human's head. This step is accomplished fast but inaccurate, while it already gets close to the target. Since it is based on a planning trajectory and an inverse modeling function in the elephant's cerebellum, it is a feedforward control step.

After getting close to the target i.e. point b, the elephant starts to move slowly and carefully until it reaches point c and places the hat. This step is slower and based on the vision of elephant, which corresponds to a feedback control step. Based on this feedforward plus feedback control scheme, the elephant can manipulate its trunk to accomplish various tasks. Note that this scheme is very similar for humans when we want to perform manipulation tasks and trajectory tracking [28] as shown in Figure 1.4.

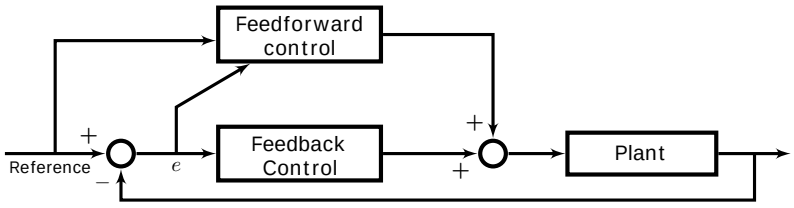


Figure 1.4: The feedback feedforward control model of cerebellum [28].

Even for an elephant that uses its trunk all its life, it remains unable to manipulate it with perfect precision using a feedforward neural system. While the bones and joints bring higher rigidity to human arms, thus we can achieve movements with high accuracy after practice. This comparison reflects the complexity of soft robot system, but also exhibits the effectiveness of this feedforward-feedback control mechanism. A natural idea is to apply such control mechanism to soft robots, while the current control performance is far from that of an elephant. The limitations and challenges are addressed in the next section.

1.3 Challenges and state of the art

The unique properties of soft robots introduce numerous complexities. On the other hand, these complexities enable the excellent performance of soft robots. New challenges are introduced at each step from design to control, which also motivates us to explore new methods.

1.3.1 Modeling

Existing first-principles about deformated body come from continuum mechanics [29]. However, it is difficult to derive control-oriented models due to the following main factors. First, the deformation of the soft robot occurs everywhere but not uniformly in the robot body, unlike the rigid robot, which deforms only at the joints. Such a system can only be formulated as an infinite-dimensional system. Second, such robots exhibit highly nonlinear dynamics caused by material properties and geometrical structures, which are further complicated by imperfect fabrication and actuation-sensing techniques [30]. Therefore, unlike the rigid case, developing reliable models in soft robotics remains challenging for both simulation and control design purposes [31].

Geometrical simplification has been exploited to build piece-wise constant curvature (PCC) models at early stage of soft robots control [16]. For PCC modeling, the soft robot is represented as a set of circular arcs with only bending behaviors being taken into account. Since the relationship between constant curvature sections can be described by a homogeneous transformation matrix [16], PCC-based methods are mostly suitable for multi-section soft manipulators. To enlarge the system application of PCC-based formulation, a dynamical model obtained by connecting the soft robot to an equivalent augmented rigid robot has been proposed in [32] for dynamic tracking control of planar soft robots. Some other PCC-based extensions, e.g., polynomial curvature modeling, have been discussed in [33]. Note that PCC-based modeling method is inherently limited to beam-like robots, which seems difficult to be adapted to a wide range of soft robot structures [34]. To overcome the difficulties in obtaining high-quality models for soft robots, data-driven and machine learning approaches have been also developed [31]. Using experimental data, these approaches directly identify the robot input-output relationship to derive accurate kinematic models for open-loop control of soft robots [35].

Although the results from continuum mechanics cannot be used directly for the control synthesis, they can be discretized for high-fidelity simulation. Two notable modeling approaches can be emphasized: i) discretized Cosserat modeling, ii) fi-

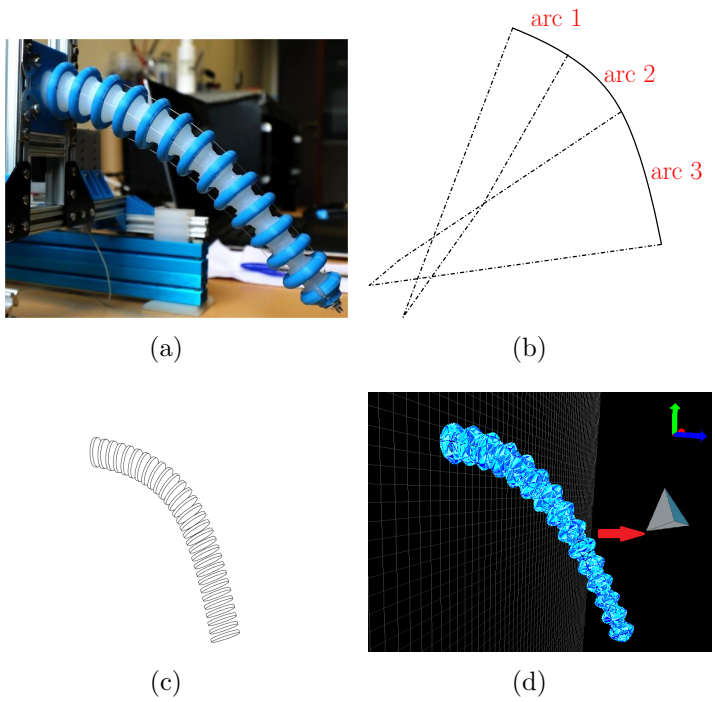


Figure 1.5: Illustration of different modeling method. (a) Real silicon rubber trunk robot. (b) Constant curvature model with three section. (c) Cosserat rod model with sliced discretization. (d) Finite element model with tetrahedron discretization.

nite element modeling. Cosserat rod theories take into account all possible rod-deformation modes, e.g., bending, twisting, shearing, stretching, under a wide range of boundary conditions [36]. The continuous Cosserat method represents the dynamics of soft robots by continuously stacking an infinite number of micro disks as shown in Figure 1.5, leading to infinite-dimensional robot models formulated as partial differential equations (PDEs). Although this method can accurately describe continuum structures in the presence of external loads and different actuation methods [2], [37], it is hard to directly leverage PDEs formulations for a control synthesis. Discretization of Cosserat models has been proposed in [38] for multisection soft manipulator dynamics under the assumption of piece-wise constant strain (PCS) along the soft arm. Inheriting from the continuous Cosserat rod theory, the discretized Cosserat method can preserve the geometrical and mechanical properties and take intro account various types of deformation and external loads. However, Cosserat-rod based modeling is currently limited to the description of multi-section soft robots with rod-like structures [38]. Moreover, a fine discretization of Cosserat-rod models

results in a drastic increase of computational burden.

Alternatively, the finite element method (FEM) is a remarkable solution for soft robot modeling, which is based on the discretization of a complex geometric robot shape into a finite number of smaller and simpler elements. Since the first FEM-based modeling and control result in [39], various extensions have been successfully developed for soft robots, e.g., collision handling [40], inflatable deformations [41], force sensing [42], etc. FEM-based methods can provide not only high-quality models but also a great flexibility to deal with a large variability of soft robot structures. Moreover, while simulating robot deformations, FEM modeling allows introducing some physical effects so that actuation methods, including magnetic field force and thermal strain, piezoelectric effect, and frictional force, can be integrated [43]. Note that FEM-based modeling has laid a solid foundation for recent theoretical and technical developments related to the SOFA[1] open-source simulation platform [41], which has proved the feasibility and the relevance of model-based strategies for soft robots.

1.3.2 Feedback control

Despite the fact that this is the core of the problem for soft robots, there are still many steps to be taken to arrive at a generic design. The main difficulties are:

1. **Lack of control oriented model**
   Existing results for controller synthesis are not directly applicable to soft robot models which are either infinite dimension or large scale and often written (and so optimized) for simulation purposes.

2. **High non-linearity from both soft material and geometrical structure**
   The proposed controller is supposed to handle drastic non-linear behavior without having a complete realistic model, or with a description of all the nonlinearities. Most results obtained in non-linear control rely on a sufficiently accurate model [44].

3. **Imperfections of actuators, sensors and robot fabrication**
   Most soft robots use hardware and actuators (cable-driven, pneumatic, artificial muscle, etc) that are manufactured with a high degree of variability in the final product. These uncertainties can neither be fully described nor properly modeled.

[1]The plugin SoftRobots of the SOFA Framework with related publications can be found at: https://project.inria.fr/softrobot.

4. **Robot configuration space is on manifold instead of Euclidean space** A soft robot is flexible but still limited, generalized coordinates for any available robot model are still constrainted on a manifold. To perform a task, it is therefore necessary to use either a motion planing method to obtain desired trajectory of generalized coordinates or to design a non-linear controller directly on the manifold.

Despite the difficulties, many new ideas have been and are proposed. The most common method is not to consider directly the dynamics of the system, but to control the robot in quasi-static mode.

Quasi-static control

Quasi-static control exploits the Jacobian matrix $J(\boldsymbol{q})$ that provides the relationship between the control input variation and the displacement of the robot end-effector. While the robot is moving at low speed, the relationship is given with:

$$\Delta \boldsymbol{y} = J(\boldsymbol{q})\Delta \boldsymbol{u}. \tag{1.1}$$

where $\boldsymbol{y}$ is the coordinate that the end-effector will reach in steady state, $\boldsymbol{u}$ is control input, $\boldsymbol{q}$ is the generalized coordinate describing robot configuration. The required control input corresponding to the desired end-effector displacement can be easily obtained with Jacobian. Since the Jacobian is a function of the robot coordinates, an additional synchronization step is required after each control execution to update the Jacobian to the new configuration, making the control quasi-static. The difference between various quasi-static control methods is mainly about how to obtain the Jacobian.

For soft robots described via the PCC model, since the relationship between the curvature of each segment of the robot and its end-effector can be described by a transformation matrix, this Jacobian matrix can be directly obtained for control. The curvature of each section in this case acts as both generalized coordinates and control input. It has been applied to several soft robots with various actuation systems and sensing techniques [16], [45]. It is almost direct to apply but limited to multi-segment robots and the constant curvature assumption is not always valid, especially for non-negligible external loads including gravity [38]. Besides the PCC model, the Jacobian matrix of the robot can also be obtained from the finite element model through the Schur complement [39]. Based on quadratic programming (QP) optimizations, a control input corresponding to the desired displacements can be obtained under possible constraints and contact conditions [41]. In order to update the Jacobian matrix, an observer built from deduced FEM model is proposed [46].

Dynamic control

The main interests of dynamic controllers lie in the following directions:

- Ensure smooth and continuous motion during robot movement
- Compensate for changes of internal and external forces: such as gravity and elastic forces
- Compensate for disturbances and uncertainties in robot motion

Although basic movements can be achieved with quasi-static controllers, it seems difficult to go further with such methods to ensure smooth motions and/or manipulations requiring dexterity. To go towards such tasks, close to real (animal) movements, a dynamic control seems to be a promising solution. Nevertheless, dynamic control synthesis requires the dynamic model of the robot, so there is few related research.

One way to achieve such a dynamic control is to enlarge the application scope of PCC-based formulation . For example, a dynamical model has been obtained by connecting the soft robot to an equivalent augmented rigid robot [32] for tracking control of planar soft robots, in which virtual inertia matrix can be estimated for each section of soft robot thus enabling dynamic control.

FEM-based dynamic feedback control looks substantially more challenging. Indeed, the quality of FEM models highly depends on the density of the mesh to be performed for the geometric robot shape. Hence, a reliable soft robot model requires a huge number of finite elements, i.e., states, which makes classical control results inapplicable. Therefore, model-order reduction is essential for dynamic feedback control design of such large-scale robot models [47]. Note that the design of FEM-based dynamic controllers is generally relied on linearized models, obtained from a linearization around either a local equilibrium [12], [48] or a desired trajectory [49]. Then, the effectiveness of these dynamic feedback controllers is generally limited to a small range of the workspace due to the large nonlinearity of soft robots. As an iterative modeling method, a global nonlinear model can not be obtained from the finite element model. However, using the framework of Linear Parameter Varying (LPV) systems or quasi-LPV gives us the possibility to represent with accuracy the nonlinear system on a large part of the workspace using a set of vertices (linear models) connected via nonlinear functions [50]. From this kind of representation, output feedback controller synthesis can be applied including performances and robustness, it is the core work of this thesis.

1.3.3 Inverse kinematics

The inverse kinematics of soft robots has not been specifically discussed, although it cannot be bypassed in the research of soft robot. Unlike rigid-body robots, most kinematic and dynamic models of soft robots do not use the same generalized coordinates. Therefore, inverse kinematics needs to be paid special attention. Nevertheless, as previously discussed, inverse kinematics can be of crucial interest when trying to complete a complex task (touching, grasping, exploring...), the target position and the robot configuration are supposed to be obtained for the task. The PCC model divides the soft robot into several idealized segments, however, such drastic model simplification also results in the robot's flexibility not being fully utilized [51]. Besides, existing data-driven models mainly focus on the robot Jacobians and endeffector instead of configurations [52]. Until now, closed form solution of inverse kinematics for soft robot has not been proposed [53].

1.4 Thesis Organization

This thesis extends the previous work of the team with Ph.D. Maxime Thieffry [48], in which the research of dynamic control for soft robots was preliminarily conducted. FEM based reduced order modeling and observer based controller design were proposed in his work, while the research is limited to linear model and small part of workspace.

The purpose of this thesis is to generalize the existing linear modeling and control framework to realize the dynamic control and planning on the larger possible subspace of the workspace, and possibly on the whole workspace.

In chapter 2, a framework for dynamic tracking control of robots is proposed. We discuss in detail the model reduction method proposed in [48] and use it to obtain the system model. This model enables the design of a feedback-feedforward controller including a disturbance observer.

In chapter 3, a generic nonlinear reduced-order tracking control method for elastic soft robots is proposed. To this end, a new linear parameter varying (LPV) control framework is developed using both reduced order models and the data collected from the soft robots.

In chapter 4, we focused on the motion planning and inverse kinematic problem of soft robots. A novel kinematic representation of soft robot combining both model and collected data is proposed, as well as a direct inverse kinematics method based on measurement data.

In the last chapter 5, we perform a preliminary research on the feed-forward

learning control method for soft robots. The idea is to enable the feedforward control of soft robot, in which reversible nonlinear models are intractable. Functional representations of feed-forward control and feed-forward learning laws with stability guarantees are illustrated. At the end, the main contributions are recalled to conclude this thesis.

The main contributions of the PhD have been the subject of the following publications:

1. Shijie Li et al., "Equivalent-Input-Disturbance Based Dynamic Tracking Control for Soft Robots Via Reduced Order Finite Element Models", IEEE/ASME Transactions on Mechatronics, vol. 27, iss. 5, pp. 4078-4089, 2022.
   This is the main material of Chapter 2.

2. Shijie Li et al., "Reduced-Order Model Based Dynamic Tracking for Soft Manipulators: Data-Driven LPV Modeling, Control Design and Experimental Results", Submitted to IFAC Control Engineering Practice.
   This is the main material of Chapter 3.

3. Shijie Li et al., "Posture Based Inverse Kinematics for Soft Manipulators", In preparation for journal publication.
   This is the main material of Chapter 4.

Chapter 2

Equivalent-Input-Disturbance-Based Dynamic Tracking Control

In the previous works of the team [48], basic framework for linear modeling and control has been proposed. Nevertheless, the resulting reduced order models are not ideal for building nonlinear models especially because the mechanical structure of the system is partially lost. In this chapter, we thoroughly studied how to use a new projector for the proper orthogonal decomposition (POD) algorithm to significantly reduce the large-scale robot models, obtained from finite element methods (FEM), while preserving their structure and stability properties. The new projector helps relocate the uncertainties of the system and the retained structure enables the design of a feedback-feedforward controller including a disturbance observer.

The feedback gains of the observer-based controller are computed from an optimization problem under linear matrix inequality constraints with guaranteed stability using Lyapunov theory. The effectiveness of the proposed dynamic control framework has been demonstrated via both high-fidelity FEM simulations and experimental validations, performed on two soft robots of different natures. In particular, comparative studies with state-of-the-art control methods have been also carried out to highlight the interests of the new soft robot control results.

2.1 Introduction

Dynamic feedback control has been proved as a solution to improve the closed-loop behaviors of soft robots with FEM models [47]. FEM models require a spatial discretization of the structure into a mesh [54], which can thus handle a large class of elastic soft robots with different geometries and materials [13]. However, the finer is the mesh the better is the model accuracy, as illustrated in Figure 2.1. Then, a reliable robot model can be only obtained with a very large number of variables. The large-scale nature of soft robot models implies technical challenges in designing dynamic controllers with conventional control tools [55]. Moreover, the obtained controllers cannot be directly applied to soft robots in practice. Hence, model order reduction (MOR) is useful for FEM-based dynamic control design [47], [56]. Regarding the existing results on MOR-based control for soft robots, the obtained models lose the structure of the mechanical system, for which the system state is composed of generalized coordinates and their derivatives. The order reduction generally does not allow to keep a structure where the state remains composed of displacement and velocities, thus losing part of the physical meaning and making motion planning more complex. Moreover, the structure mixing displacements and velocities spreads the uncertainties in a more important way and thus introduces an additional conservatism in the conditions to get a robust controller.

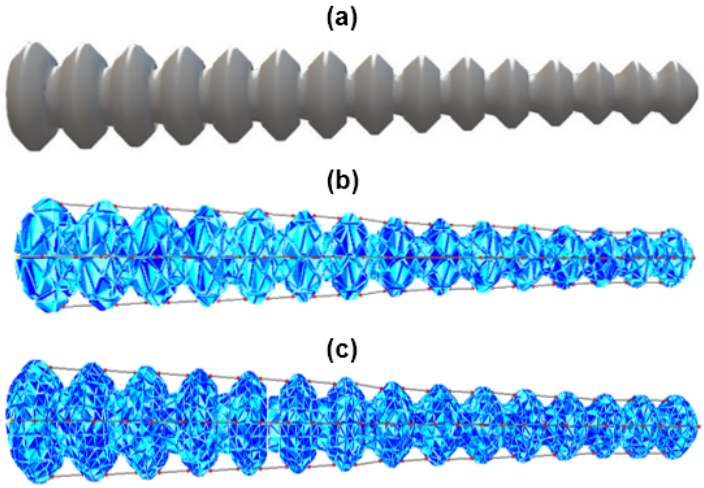


Figure 2.1: Illustrations of FEM modeling for a Trunk robot. (a) Visual model. (b) FEM model with a coarse mesh. (c) FEM model with a medium-size mesh. The blue triangulars on the FEM model are the different facets of the tetrahedral elements.

Motivated by the above technical issues, we modify the proper orthogonal decomposition (POD) method used in [47], [48] to significantly reduce the order of FEM soft robot models while preserving their structure and stability properties. As it will be shown, being able to preserve the structure enables a more effective dynamic control scheme, especially in case of uncertainty and disturbances. The proposed control scheme is formed with three main components, i.e., feedforward control, disturbance-estimator control and feedback control. Using the dynamic FEM reduced-order model, the feedforward action is designed to account for the reference trajectory whereas the disturbance-estimator control is designed to compensate the modeling uncertainty. The observer-based feedback control is designed to guarantee a desired tracking performance, specified by an optimized $\ell_\infty-$gain and a predefined linear matrix inequality (LMI) region of the closed-loop robot system. Specifically, our main contributions can be summarized as follows:

- Compared to the related works [47], [48], we propose a new projector for the POD reduction method. Then, the reduced-order models of soft robots can be obtained while preserving the structure and stability properties, which is crucial to design an effective feedback-feedforward control scheme in presence of modeling uncertainties.

- Exploiting the equivalent-input-disturbance (EID) concept [57], [58] for a specific mechanical model structure of soft robots, we develop a FEM model-based control framework to achieve high-precision tracking tasks despite of unknown uncertainties and disturbances. The closed-loop performance is rigorously guaranteed by Lyapunov stability theory, which is advantage in comparison with most of existing results on soft robotics control.

- High-fidelity SOFA simulations and experimental validations have been performed on two soft robots with different natures to demonstrate the effectiveness of the proposed control framework.

2.2 Modeling of Soft Robots

This section provides a procedure to obtain control-based models for large-scale soft robotics systems.

2.2.1 From FEM Modeling to State-Space Representation

For the modeling of soft robots, FEM method is used to approximate the infinite-dimensional model by subdividing it into a large amount of tiny elements [54]. The resulting discretized model has the number of degrees of freedom proportional to the number of elements. The dynamics of a deformable soft robot can be described as follows [41]:

$$\mathcal{M}(q)\ddot{q} = \mathcal{P}(q) - \mathcal{F}(q,\dot{q}) + \mathcal{H}(q)u, \tag{2.1}$$

where $q \in \mathbb{R}^n$ is the displacement vector, $\dot{q} \in \mathbb{R}^n$ is the velocity vector, $u \in \mathbb{R}^m$ is the control input, $\mathcal{M}(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\mathcal{F}(q,\dot{q})$ represents the internal elastic forces of soft robots, $\mathcal{P}(q)$ represents known external forces, $\mathcal{H}(q)$ is the control input matrix. We consider the case that $\mathcal{P}(q)$ only contains the gravity force, and the mass distribution does not change over time. Hence, the positive definite matrices $\mathcal{P}(q) = P$ and $\mathcal{M}(q) = M$ are constant. Without loss of generality, assume that the tracking control of soft robots is performed around an equilibrium point, defined as $(q_0, \dot{q}_0) = (0, 0)$ and $u_0 = 0$. Then, it follows from (2.1) that

$$P - \mathcal{F}(0,0) = 0. \tag{2.2}$$

Moreover, around the equilibrium point, the internal force $\mathcal{F}(q,\dot{q})$ can be approximated as

$$\mathcal{F}(q,\dot{q}) \approx \mathcal{F}(0,0) + \mathcal{K}(q,\dot{q})q + \mathcal{D}(q,\dot{q})\dot{q}. \tag{2.3}$$

The compliance matrix $\mathcal{K}(q,\dot{q})$ and the damping matrix $D(q,\dot{q})$ are respectively defined as

$$\mathcal{K}(q,\dot{q}) = \frac{\partial \mathcal{F}(q,\dot{q})}{\partial q}, \quad \mathcal{D}(q,\dot{q}) = \frac{\partial \mathcal{F}(q,\dot{q})}{\partial \dot{q}}.$$

Substituting (2.2) and (2.3) into (2.1), the linearized FEM model of the soft robot can be obtained as

$$M\ddot{q} \approx -\mathcal{K}(0,0)q - \mathcal{D}(0,0)\dot{q} + \mathcal{H}(0)u. \tag{2.4}$$

For conciseness, we denote $K = \mathcal{K}(0,0)$, $D = \mathcal{D}(0,0)$ and $H = \mathcal{H}(0)$. Then, system (2.4) can be rewritten in the state-space form

$$\dot{x} = Ax + Bu, \quad y = Cx, \tag{2.5}$$

where $x = \begin{bmatrix}\dot{q}^\top & q^\top\end{bmatrix}^\top \in \mathbb{R}^{2n}$. The matrices $A \in \mathbb{R}^{2n\times 2n}$ and $B \in \mathbb{R}^{2n\times m}$ are large-scale sparse, defined as

$$A = \begin{bmatrix} -M^{-1}D & -M^{-1}K \\ I & 0 \end{bmatrix}, \quad B = \begin{bmatrix} M^{-1}H \\ 0 \end{bmatrix}. \tag{2.6}$$

The system output $y \in \mathbb{R}^p$ represents the coordinates of the robot end-effector.

Remark 1. The compliance matrix K and the damping matrix D are symmetric positive definite. Due to the large-scale feature of system (2.5), i.e., $n > 3000$, and the scattered connection between local neighbor elements in FEM modeling, these matrices are also sparse.

2.2.2 Model Order Reduction for Soft Robots

Many model order reduction methods have been proposed for large-scale systems, e.g., singular value decomposition (SVD), moment matching [59]. As an SVD-based method, POD enables an effective model order reduction with a priori error bound. Moreover, in contrast to most of existing order reduction methods, POD algorithms can be directly applied to large-scale nonlinear systems [60]. Hence, POD has been shown as a suitable order reduction method for soft robots [47].

At the beginning of the POD process, responses of the system states with respect to excitation signals are stacked to construct snapshot matrix. To exploit the data interdependence, the SVD of snapshot matrix is performed to obtain the modal decomposition of system response. The system state can be significantly reduced through an orthogonal projection operator $T_r^* \in \mathbb{R}^{2n\times 2l}$, defined as

$$\begin{bmatrix} x_r \\ x_{\bar{r}} \end{bmatrix} = \begin{bmatrix} T_r^* \\ T_{\bar{r}}^* \end{bmatrix} x, \tag{2.7}$$

where $x_r \in \mathbb{R}^{2l}$ is the reduced state vector with $l \ll n$, $x_{\bar{r}}$ is the neglected state vector, T_r^* is the projector truncated from left singular matrix of SVD and $T_{\bar{r}}^*$ is the orthogonal complement of matrix T_r^*. Since the POD method only requires SVD operations to obtain the projector, this method is computationally efficient for a priori given snapshots [60].

Despite its effectiveness in reducing the order of system (2.5) for control design, the POD projection (2.7) generally does not allow preserving the stability and the mechanical structure properties of this large-scale system [61], which are crucial for the proposed EID-based control scheme. To overcome this drawback, we propose a modified POD projection taking into account the specific matrix structures of system (2.5). To this end, only the displacement vector is projected into the low-dimension space. Then, the reduced velocity vector can be obtained with the same projector matrix. As a result, we have

$$x_r = \begin{bmatrix} \dot{q}_r \\ q_r \end{bmatrix} = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} \dot{q} \\ q \end{bmatrix} = T_r x. \tag{2.8}$$

Applying the proposed projector (2.8) to system (2.5), the following approximated reduced-order model can be obtained:

$$\begin{aligned} \dot{x}_r &= A_r x_r + B_r u + T_r A T_{\bar{r}}^{\top} x_{\bar{r}}, \\ y &= C_r x_r, \end{aligned} \tag{2.9}$$

where $T_{\bar{r}}$ is the orthogonal complement of matrix T_r, and

$$\begin{aligned} A_r &= T_r A T_r^{\top} = \begin{bmatrix} -TM^{-1}DT^{\top} & -TM^{-1}KT^{\top} \\ I & 0 \end{bmatrix}, \\ B_r &= T_r B = \begin{bmatrix} TM^{-1}H \\ 0 \end{bmatrix}, \quad C_r = CT_r^{\top}. \end{aligned} \tag{2.10}$$

The accuracy of the POD reduced-order models depends on the decay rate of the singular values of the snapshots [47]. Figure 2.2(a) depicts the evolution of the singular values corresponding to the position snapshots of the Trunk robot discussed in Section 2.5. As shown in Figure 2.2(b), a fast decay of singular values is clearly observed for the first four values, which represent more than 95% of the singular values of the 3324-state FEM robot model. Then, the POD method with the proposed projector (2.8) can significantly reduce the number of the state variables, i.e., from 3324 to 4 states, while keeping a good modeling quality for dynamic control purposes.

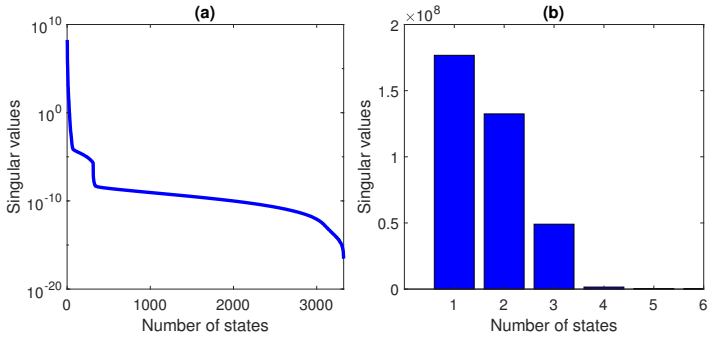


Figure 2.2: Singular values of the position snapshots of a Trunk robot. (a) Evolution of the singular values. (b) Six first singular values.

Remark 2. With the proposed POD projector (2.8), the matrix structures are preserved between system (2.6) and its reduced-order counterpart (2.9), i.e., the parameters are only involved in the upper-half of the state-space matrices. As shown in the sequel, this enables an effective EID-based control framework to compensate

the modeling uncertainty for tracking performance improvements. This model order reduction property preservation has not been yet exploited for dynamic control of soft robots.

Model Error Analysis

We consider uncertainty in the knowledge of compliance, damping and control matrices as

$$\hat{K} = K + \Delta K, \quad \hat{D} = D + \Delta D, \quad \hat{H} = H + \Delta H,$$

where $\hat{K}$, $\hat{D}$, $\hat{H}$ are the estimations, K, D, H the real values, and ΔK, ΔD, ΔH the uncertainties. Since the system structure is preserved, these uncertain terms are taken into account in the reduced-order model (2.9) as

$$\dot{x}_r = \hat{A}_r x_r + \hat{B}_r u + T_r \hat{A} T_{\bar{r}}^\top x_{\bar{r}}, \tag{2.11}$$

with

$$\begin{aligned}
&\hat{A}_r = A_r + \Delta A_r, \quad \hat{B}_r = B_r + \Delta B_r, \\
&\hat{A} = A + \Delta A, \qquad \Delta A = \begin{bmatrix} -M^{-1}\Delta D & -M^{-1}\Delta K^\top \\ 0 & 0 \end{bmatrix}, \\
&\Delta A_r = T_r \Delta A T_r^\top, \quad \Delta B_r = \begin{bmatrix} T M^{-1} \Delta H \\ 0 \end{bmatrix}.
\end{aligned}$$

Note that due to the specific upper-half structures of matrices B_r, ΔA_r and ΔB_r, the following matrix decompositions can be performed:

$$\Delta A_r = B_r B_r^\dagger \Delta \bar{A}_r, \quad \Delta B_r = B_r B_r^\dagger \Delta \bar{B}_r, \tag{2.12}$$

where $B_r^\dagger = \left(B_r^\top B_r\right)^{-1} B_r^\top$ is the pseudo-inverse of B_r, and

$$\begin{aligned}
&\Delta \bar{A}_r = \begin{bmatrix} -TM^{-1}\Delta D T^\top & -TM^{-1}\Delta K T^\top \end{bmatrix}, \\
&\Delta \bar{B}_r = TM^{-1}\Delta H.
\end{aligned}$$

Let us define a lumped disturbance as

$$d_l = B_r^\dagger \Delta \bar{A}_r x_r + B_r^\dagger \Delta \bar{B}_r u. \tag{2.13}$$

Moreover, to take into account the neglected state dynamics for the control design, we can decompose

$$T_r \hat{A} T_{\bar{r}}^\top x_{\bar{r}} = B_r d_{\ddot{q}} + B_{\bar{r}} d_{\dot{q}}. \tag{2.14}$$

where $B_{\bar{r}}$ is an orthogonal complement of B_r. Note that the disturbance $d_{\ddot{q}}$ (respectively $d_{\dot{q}}$) corresponds to the neglected dynamics related to the acceleration $\ddot{q}$ (respectively velocity $\dot{q}$). From (2.12), (2.13) and (2.14), the uncertain reduced-order model (2.11) can be represented as

$$\begin{aligned} \dot{x}_r &= A_r x_r + B_r(u + d_e) + B_{\bar{r}} d_{\dot{q}}, \\ y &= C_r x_r, \end{aligned} \tag{2.15}$$

with $d_e = d_{\ddot{q}} + d_l$.

Transformation of Generalized Coordinates

Although the proposed POD projector (2.8) allows preserving the structures of the state-space matrices, the state variables of the reduced-order model (2.9) are not the displacement and the velocity of soft robots in the Cartesian coordinates. These new state variables can be considered as generalized coordinates. Hence, a coordinate transformation between generalized coordinates and Cartesian coordinates is required to design an effective model-based feedforward control action. Indeed, for trajectory tracking only the robot outputs are required to track their corresponding reference signals, directly defined in the Cartesian coordinate system, i.e., a reference model is generally not available in the generalized coordinate system for control design as in [48].

Since the output of soft robots only includes the coordinates of the end-effectors, it follows from (2.9) that

$$y = C_r x_r = C_g q_r,$$

with $C_r = \begin{bmatrix} 0 & C_g \end{bmatrix}$ and $C_g \in \mathbb{R}^{p \times l}$. Then, if the order reduction is performed with $p = l$, then the proposed POD method can provide a full-rank matrix C_g. Hence, for any arbitrary desired trajectory r in Cartesian coordinates, the corresponding trajectory r_g in generalized coordinates can be defined as $r_g = C_g^{-1} r$. As shown in (2.18), this relation allows for a full use of the dynamic FEM reduced-order model (2.15) of soft robots to design an effective feedforward control action.

2.3 Tracking Control Problem Formulation

This section formulates the tracking control problem of soft robots. To this end, we define the tracking error in generalized coordinates as $e = x_r - r_g$. Then, the tracking error dynamics can be defined from (2.15) as

$$\dot{e} = A_r e + B_r(u + d_e) + A_r r_g - \dot{r}_g + B_{\bar{r}} d_{\dot{q}}. \tag{2.16}$$

To deal with the modeling uncertainty and improve the tracking control performance of soft robots, we propose a feedback-feedforward control scheme composed of three components as

$$u = u_{ff} + u_{dc} + u_{fb}, \tag{2.17}$$

where u_{ff} is the feedforward control, u_{dc} is the disturbance-estimator based control, and u_{fb} is the feedback control. The proposed tracking control scheme is illustrated in Figure 2.3.

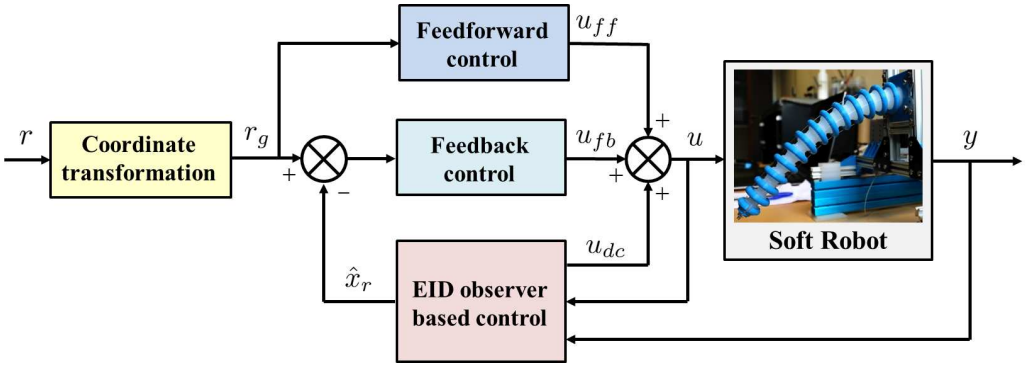


Figure 2.3: EID observer-based tracking control structure for soft robots.

Feedforward Control

The feedforward control u_{ff} accounts for the affect of the reference signal r_g, considered as a known disturbance, on the closed-loop system. Hence, this control action is designed such that

$$B_r u_{ff} = -(A_r r_g - \dot{r}_g). \tag{2.18}$$

The feedforward action can be designed from (2.18) as

$$u_{ff} = -B_r^{\dagger}(A_r r_g - \dot{r}_g). \tag{2.19}$$

Disturbance-Estimator Based Control

The lumped disturbance d_e includes the nonlinearity and modeling uncertainty. Note from (2.16) that d_e enters in the system dynamics via the same channel as the

input u, i.e., matching disturbance. The estimation of disturbance $\hat{d}_e$ consists of high-frequency noise, so that a filtered disturbance signal $\hat{d}_{ef}$ is used for the compensation. Then, the disturbance-estimator based control can be designed as follows:

$$u_{dc} = -\hat{d}_{ef}, \tag{2.20}$$

where $\hat{d}_{ef}$ is an estimate of the filtered signal d_{ef} of the lumped disturbance d_e. To estimate this disturbance, we assume that d_e is of low-frequency, whose dynamics can be efficiently captured using a second-order polynomial signal [62]. Note that this assumption is suitable due to the low-frequency behaviors of soft robots. Moreover, inspired by the EID approach [57], we integrate a low-pass filter with a time constant T_f of the form

$$\mathbf{F}(s) = \frac{1}{1 + T_f s} I, \tag{2.21}$$

where s is the Laplace variable, to limit the angular-frequency band of the disturbance estimate as shown in Figure 2.3. Then, the disturbance model is given as

$$\dot{d} = Jd, \tag{2.22}$$

with

$$d = \begin{bmatrix} d_{ef} \\ d_e \\ \dot{d}_e \end{bmatrix}, \quad J = \begin{bmatrix} -\frac{1}{T_f} I & \frac{1}{T_f} I & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix}.$$

From (2.15) and (2.22), we propose the following Luenberger-like unknown-input observer to estimate simultaneously the state x_r and the unknown disturbance d_e:

$$\begin{aligned} \dot{\hat{x}}_r &= A_r \hat{x}_r + B_r(u + \hat{d}_e) + L_x(y - \hat{y}), \\ \dot{\hat{d}} &= J\hat{d} + L_d(y - \hat{y}), \\ \hat{y} &= C_r \hat{x}_r, \end{aligned} \tag{2.23}$$

where $\hat{d}$ is the estimate of d, the observer gains $L_x \in \mathbb{R}^{2l \times p}$ and $L_d \in \mathbb{R}^{3m \times p}$ are to be designed.

Remark 3. The filter $\mathbf{F}(s)$ in (2.21) is integrated into the PI observer (2.23) to offer a degree of freedom to regulate the angular-frequency band for disturbance rejection. Then, the value of T_f can be selected as

$$T_f \in \left[\frac{1}{10\omega_r}, \frac{1}{5\omega_r} \right],$$

where ω_r is the highest angular frequency selected for disturbance rejection purposes [57].

The dynamics of estimation error $\varepsilon = \begin{bmatrix} x_r - \hat{x}_r \\ d - \hat{d} \end{bmatrix}$ can be defined from (2.15), (2.22) and (2.23) as follows:

$$\dot{\varepsilon} = (A_o - L_o C_o)\varepsilon + B_{od} d_{\dot{q}}, \tag{2.24}$$

with

$$A_o = \begin{bmatrix} A_r & \tilde{B}_r \\ 0 & J \end{bmatrix}, \quad B_{od} = \begin{bmatrix} B_{\bar{r}} \\ 0 \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} x_r - \hat{x}_r \\ d - \hat{d} \end{bmatrix},$$
$$L_o = \begin{bmatrix} L_x^\top & L_d^\top \end{bmatrix}^\top, \quad \tilde{B}_r = \begin{bmatrix} B_r & 0 \end{bmatrix}, \quad C_o = \begin{bmatrix} C_r & 0 \end{bmatrix}.$$

Feedback Control

The feedback control aims at guaranteeing the closed-loop stability and improving the steady-state tracking performance. To this end, we consider the following proportional-integral control law:

$$u_{fb} = K_p(\hat{x}_r - r_g) + K_i e_i, \tag{2.25}$$

where $K_p \in \mathbb{R}^{m \times 2l}$ and $K_i \in \mathbb{R}^{m \times p}$ are the feedback gains to be designed, and

$$\dot{e}_i = C_r(\hat{x}_r - r_g). \tag{2.26}$$

With u_{ff}, u_{dc} and u_{fb} respectively defined in (2.19), (2.20) and (2.25), substituting the control expression (2.17) into system (2.16), the tracking error dynamics can be represented as

$$\dot{\xi} = (A_c + B_c K_c)\xi + B_o \varepsilon + B_{cd} d_{\dot{q}}, \tag{2.27}$$

where

$$\xi = \begin{bmatrix} e^\top & e_i^\top \end{bmatrix}^\top, \quad A_c = \begin{bmatrix} A_r & 0 \\ C_r & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} B_r \\ 0 \end{bmatrix},$$
$$K_c = \begin{bmatrix} K_p & K_i \end{bmatrix}, \quad B_o = \begin{bmatrix} -B_r K_p & 0 \\ 0 & 0 \end{bmatrix}, \quad B_{cd} = \begin{bmatrix} B_{\bar{r}} \\ 0 \end{bmatrix}.$$

The extended closed-loop system can be defined from (2.24) and (2.27) as

$$\dot{\tilde{x}} = \begin{bmatrix} \bar{A}_c & B_o \\ 0 & \bar{A}_o \end{bmatrix} \tilde{x} + \begin{bmatrix} B_{cd} \\ B_{od} \end{bmatrix} d_{\dot{q}}, \tag{2.28}$$

where $\tilde{x} = \begin{bmatrix} \xi^\top & \varepsilon^\top \end{bmatrix}^\top$, and

$$\bar{A}_c = A_c + B_c K_c, \quad \bar{A}_o = A_o - L_o C_o.$$

Since we focus on the tracking control, the performance output associated to system (2.28) is the position tracking error, i.e., $z = e = \tilde{C}_r \xi$, with $\tilde{C}_r = \begin{bmatrix} C_r & 0 \end{bmatrix}$. Hence, one has

$$z = F\tilde{x}, \quad F = \begin{bmatrix} \tilde{C}_r & 0 \end{bmatrix}. \tag{2.29}$$

To specify the performance of the closed-loop system, we exploit the concept of $\mathscr{D}-$stability [63] to design both the observer (2.23) and the feedback controller (2.25). The idea of $\mathscr{D}-$stability is to design a controller or observer whose poles are located in specific regions of a complex plane.

To this end, we consider an LMI region $\mathscr{D}(r, \theta, \alpha)$ defined as a subset of the left-half complex plane to guarantee a minimum decay rate α, a minimum damping ratio $\zeta = \cos(\theta)$ and a maximum undamped natural frequency $\omega_d = r\sin(\theta)$. The following lemma guarantees the $\mathscr{D}-$stability of a matrix $\mathbf{A}$.

Lemma 1 ([63])**.** All the eigenvalues of $\mathbf{A}$ are located inside the region $\mathscr{D}(r, \theta, \alpha)$ if and only if there exists a symmetric positive definite matrix $\mathbf{X}$ such that

$$\begin{aligned}
&\mathbf{A}^\top \mathbf{X} + \mathbf{X}\mathbf{A} + 2\alpha \mathbf{X} \prec 0, \\
&\begin{bmatrix} -r\mathbf{X} & \mathbf{A}\mathbf{X} \\ \star & -r\mathbf{X} \end{bmatrix} \prec 0, \\
&\begin{bmatrix} \sin(\theta)\left(\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^\top\right) & \cos(\theta)\left(\mathbf{A}\mathbf{X} - \mathbf{X}\mathbf{A}^\top\right) \\ \star & \sin(\theta)\left(\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^\top\right) \end{bmatrix} \prec 0.
\end{aligned}$$

Hereafter, we propose an effective solution for the following observer-based feedback control problem.

Problem 1. Consider two LMI regions $\mathscr{D}_c(r_c, \theta_c, \alpha_c)$ and $\mathscr{D}_o(r_o, \theta_o, \alpha_o)$. Determine the control gain K_c (respectively the observer gain L_o) such that the poles of the dynamic matrix $\bar{A}_c$ (respectively $\bar{A}_o$) remain inside the region $\mathscr{D}_c(r_c, \theta_c, \alpha_c)$ (respectively $\mathscr{D}_o(r_o, \theta_o, \alpha_o)$). Moreover, the extended system (2.28) satisfies the following closed-loop properties.

(P1) For $d_{\dot{q}}(t) = 0$, $\forall t \geq 0$, system (2.28) is exponentially stable with a decay rate $\alpha > 0$. For any initial condition $\tilde{x}(0)$ and any bounded disturbance $d_{\dot{q}}(t) \in \mathscr{B}_\infty$, there exists a bound $\eta(\tilde{x}(0), \|d_{\dot{q}}\|_\infty)$ such that

$$\tilde{x}(t) \leq \eta(\tilde{x}(0), \|d_{\dot{q}}\|_\infty), \quad t \geq 0.$$

(P2) For $\forall \tilde{x}(0)$ and $\forall d_{\dot{q}}(t) \in \mathscr{B}_\infty$, we have performance output z bounded as

$$\limsup_{t\to\infty} \|z\| \leq \gamma \|d_{\dot{q}}\|_\infty, \quad \gamma > 0, \tag{2.30}$$

where the $\ell_\infty-$gain is specified in Theorem 1.

Remark from (2.29) and (2.30) that a smaller value of the $\ell_\infty-$gain γ leads to a better tracking control performance.

2.4 EID-Based Output Feedback Tracking Control

This section provides conditions to design both the unknown input observer (2.23) and the feedback control law (2.25) satisfying the closed-loop specifications in Problem 1.

Theorem 1. Consider the closed-loop system (2.28) and two LMI regions $\mathscr{D}_c(r_c, \theta_c, \alpha_c)$ and $\mathscr{D}_o(r_o, \theta_o, \alpha_o)$. If there exist symmetric positive definite matrices $X \in \mathbb{R}^{(2l+p)\times(2l+p)}$, $Q \in \mathbb{R}^{(2l+3m)\times(2l+3m)}$, matrices $M \in \mathbb{R}^{m\times(2l+p)}$, $N \in \mathbb{R}^{(2l+3m)\times p}$, and positive scalars μ, ν such that the optimization problem (2.31) is feasible.

$$\text{minimize} \qquad \mu + \nu, \tag{2.31}$$

$$\text{subject to} \tag{2.32}$$

$$\mathrm{He}\begin{bmatrix} \mathscr{A}_o + \alpha_c Q & QB_{od} \\ 0 & -\alpha_c \nu I \end{bmatrix} \prec 0, \tag{2.33}$$

$$\mathrm{He}(\mathscr{A}_c + \alpha_c X) \prec 0, \tag{2.34}$$

$$\begin{bmatrix} -r_c X & \mathscr{A}_c \\ \star & -r_c X \end{bmatrix} \prec 0, \tag{2.35}$$

$$\begin{bmatrix} \sin(\theta_c)\mathrm{He}(\mathscr{A}_c) & \cos(\theta_c)\mathrm{Sym}(\mathscr{A}_c) \\ \star & \sin(\theta_c)\mathrm{He}(\mathscr{A}_c) \end{bmatrix} \prec 0, \tag{2.36}$$

$$\mathrm{He}(\mathscr{A}_o + \alpha_o Q) \prec 0, \tag{2.37}$$

$$\begin{bmatrix} -r_o Q & \mathscr{A}_o \\ \star & -r_o Q \end{bmatrix} \prec 0, \tag{2.38}$$

$$\begin{bmatrix} \sin(\theta_o)\mathrm{He}(\mathscr{A}_o) & \cos(\theta_o)\mathrm{Sym}(\mathscr{A}_o) \\ \star & \sin(\theta_o)\mathrm{He}(\mathscr{A}_o) \end{bmatrix} \prec 0, \tag{2.39}$$

$$\begin{bmatrix} X & 0 & X\tilde{C}_r^\top \\ \star & Q & 0 \\ \star & \star & \mu I \end{bmatrix} \succeq 0, \tag{2.40}$$

with

$$\mathscr{A}_c = A_c X + B_c M, \quad \mathscr{A}_o = QA_o - NC_o.$$

Then, the feedback observer-based controller (2.25) is such that the closed-loop properties specified in Problem 1 are satisfied with a guaranteed $\ell_\infty-$gain $\gamma = \sqrt{\nu\mu}$. Furthermore, the control and observer gains are respectively given by

$$K_c = MX^{-1}, \quad L_o = Q^{-1}N. \tag{2.41}$$

Proof. The proof is postponed to Appendix A. □

Remark 4. The design procedure in Theorem 1 is recast as a convex optimization problem under LMI constraints. Here, the control gain K_c and observer gain L_o can be effectively computed using YALMIP toolbox and SeDuMi solver [64].

Remark 5. The LMI region $\mathscr{D}_c(r_c, \theta_c, \alpha_c)$ is defined to represent the dominant low-frequency dynamics of soft robots. Moreover, the damping ratio should be chosen as small as possible to avoid oscillatory behaviors of the closed-loop robot systems. However, without restrictive constraints, the LMI region $\mathscr{D}_o(r_o, \theta_o, \alpha_o)$ should be specified to guarantee a fast convergence of the estimation error dynamics (2.24).

2.5 Illustrative Results and Evaluations

This section presents the control results obtained with both FEM-based simulations and realtime experiments. To highlight the systematic feature of the proposed tracking control method, we consider two different silicone soft robots for validations: Diamond robot and Trunk robot, whose physical prototypes are described in [65].

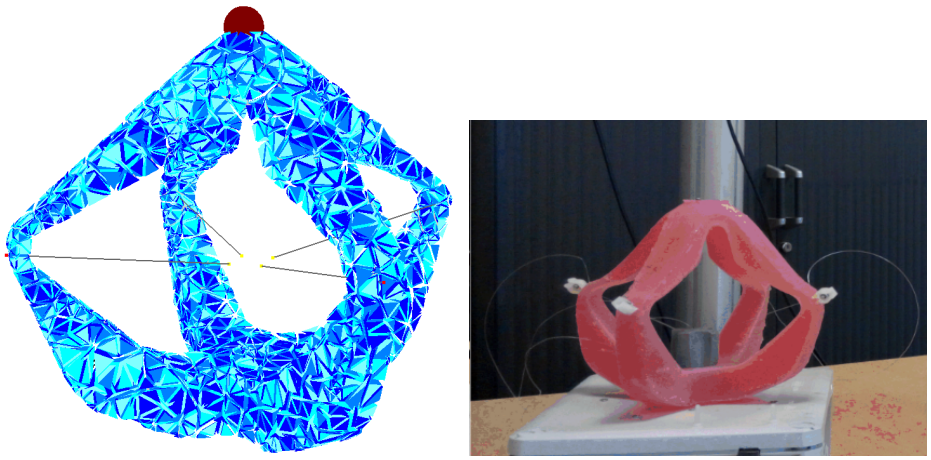
2.5.1 FEM-Based Validations with Diamond Robot

The Diamond robot has four soft legs, which are actuated by cables as depicted in Figure 2.4. This cable-driven robot can be considered as a soft version of parallel robots. The movement of the Diamond robot is realized through a combination of bending and compression of its four legs. Hence, most of the PCC-based approaches [33] are no longer suitable for dynamic modeling and control of such a soft robot. The weight of the Diamond robot is 0.5 [kg] and its height is 110 [mm] in the initial position. The material parameters of this soft robot are obtained through experiments. Then, the FEM model of the Diamond robot can be derived with 1570 nodes, leading to 9420 state variables. This high-fidelity FEM model is implemented in the SOFA open-source simulation platform [41] for validations. Note that the modeling error between FEM-based model and the real robot is less than 10% in a workspace of $40 \times 40 \times 20$ [mm^3] around the rest position of the robot [39]. For control design, a six-order dynamical model of the Diamond robot can be obtained with the proposed POD model reduction method.

The FEM simulation is conducted in the SOFA framework. SOFA is an open source framework dedicated to research, prototyping and development of physics-based simulations. Sofa especially has advantages in real-time simulation of soft and

elastic objects, so it is applied to soft robotics and surgical simulation. Our partner DEFROST team in INRIA LILLE HAS been contributing to the research of soft robotics and the development of the soft robotics toolbox in SOFA. More details on the plugin SoftRobots for the SOFA Framework with related publications can be found at the address: https://project.inria.fr/softrobot.

The control algorithms are implemented in MATLAB/SIMULINK environment while the actuation is simulated in SOFA through TCP/IP socket. This SIMULINK-SOFA co-simulation environment allows reducing significantly the costs related to not only the design but also the real-time validation of complex soft robots.



(a) (b)

Figure 2.4: FEM-based schematic and real platform of the Diamond robot. The red point in (a) represents the end-effector and the gray lines represent the driving cables. These cables are shown in (b).

For the tracking control of the Diamond robot, we consider end effector trajectories of the form

$$y_x(t) = 15\cos(\omega t),\ y_y(t) = 15\sin(\omega t),\ y_z(t) = 110 + t,\ t \in [0, 20] \tag{2.42}$$

with $\omega = 1$ [rad/s] for a fast trajectory tracking, and $\omega = 0.33$ [rad/s] for a slow trajectory tracking. This trajectory is defined within the cylindrical workspace of the Diamond robot. To evaluate the control robustness performance with respect to the modeling errors, we consider two test scenarios, i.e., without and with uncertainty on Young's modulus. This latter characterizes the softness property of the silicone material, i.e., a softer material leads to a stronger oscillatory response in the open

loop. Moreover, a comparison between the following FEM-based control methods is performed to demonstrate the effectiveness of the new control method:

- Method 1: Inverse kinematics based QP control [15], [39].
- Method 2: Jacobian-based PID control [66], [67].
- Method 3: Proposed EID-based control.

For Method 1, the QP-based controller is composed of an PI controller and an QP solver. The PI controller aims at providing a desired action of the robot end-effector whereas the QP solver computes the actuation control through an inverse kinematic optimization. The details on the QP-based control architecture and the related tuning methods can be found in [15].

For Method 2, the PID controller is defined as

$$u(t) = \mathscr{J}^{-1}\left(K_p e(t) + K_i \int e(t)dt + K_d \dot{e}(t)\right),$$

where $\mathscr{J}$ is the Jacobian matrix of the soft robot, and K_p, K_i, K_d are the PID control gains. Note that the Jacobian matrices of soft robots can be directly obtained with SOFA platform for control design. Then, the PID controllers designed in both Method 1 and Method 2 are properly tuned. Note also that due to the low resonant frequency characteristics of soft robots, the ranges of the PID control gains are quite narrow to avoid aggressive closed-loop behaviors.

Scenario 1 [Without Uncertainty on Young's Modulus]

For this test scenario, the Young's modulus of the FEM simulation model and the FEM control-based model of the Diamond robot are both set to 150 [KPa], i.e., no mismatch between two FEM models. The slow trajectory tracking is shown in Figure 2.5. The feedforward action of the proposed controller and the inverse kinematic control of Method 1 allow for a better tracking performance during the steady-state phase compared to the PID controller. Observe in Figure 2.6 that the drawbacks of Methods 1 and 2 become much more clear with a fast trajectory tracking in comparison to the EID-based method, i.e., the dynamic response of the QP-based controller is degraded and the steady-state tracking error induced by the PID controller increases.

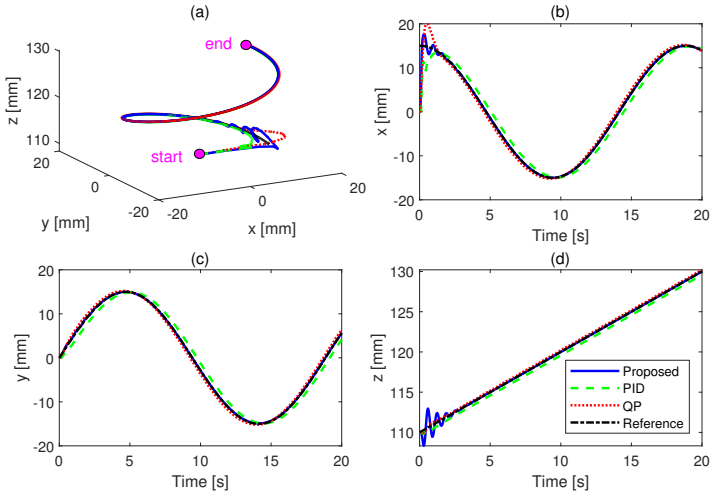


Figure 2.5: Slow spiral trajectory tracking in Scenario 1.

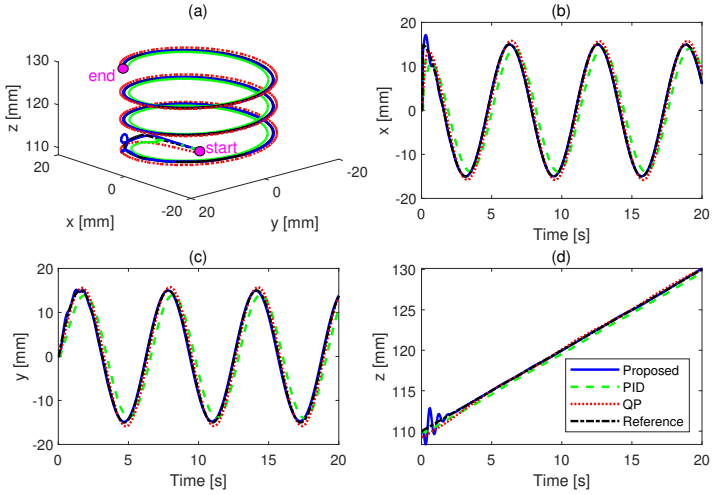


Figure 2.6: Fast spiral trajectory tracking in Scenario 1.

Scenario 2 [With Uncertainty on Young's Modulus]

The Young's modulus of the Diamond robot implemented in SOFA platform remains 150 [KPa] as in Scenario 1. However, we modify the Young's modulus of the FEM control-based robot model to 180 [KPa] to introduce a modeling uncertainty in the control design. The control performance of the three considered methods for a slow spiral trajectory tracking is depicted in Figure 2.7. Due to modeling

errors and without taking into account the robot dynamics, as expected the QP-based controller leads to a worse behavior in transient responses compared to the tracking results in Scenario 1. Note that a satisfactory control performance can be maintained for both PID controller and EID-based controller in case of a slow trajectory tracking. Figure 2.8 presents the fast trajectory tracking results. Since the modeling error directly affects the Jacobian matrix and the inverse kinematics, the tracking performance of the PID controller and QP controller are degraded, whereas the closed-loop behavior is preserved with the proposed controller due to its effective EID-based error compensation mechanism.

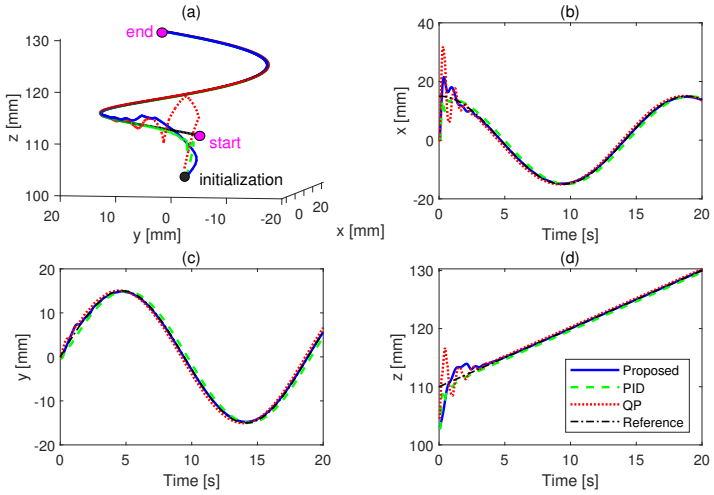


Figure 2.7: Slow spiral trajectory tracking in Scenario 2.

For a quantitative performance analysis, we define the following integral square tracking error (ISE) index:

$$\text{ISE} = \int_0^{\Delta t} e(t)^2 dt, \tag{2.43}$$

where Δt is the tracking time. Figure 2.9 summarizes the tracking performance comparisons in terms of ISE index [mm^{2}s] between the three FEM-based control methods obtained with the 3D trajectory (2.42) and $\Delta t = 20$ [s]. Remark that when there is no significant modeling uncertainty, the QP-based controller can provide a better tracking performance compared to the Jacobian-based PID controller. However, this latter seems to be more robust with respect to uncertainties than the QP-based controller. We can see also that the proposed EID-based control method provides the best tracking performance in all the test scenarios.

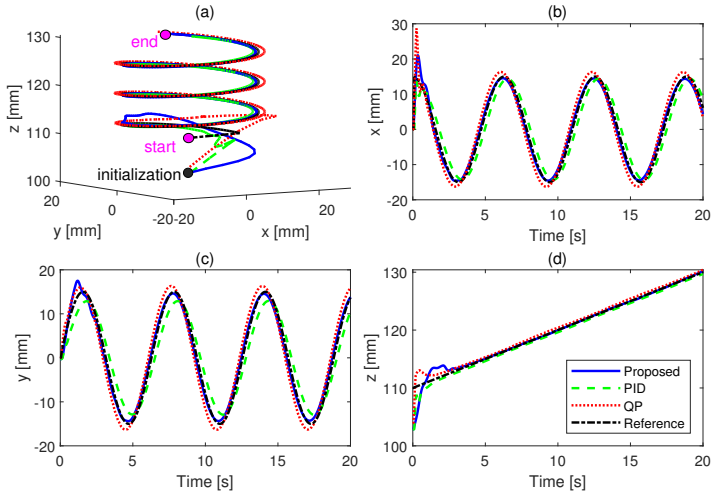


Figure 2.8: Fast spiral trajectory tracking in Scenario 2.

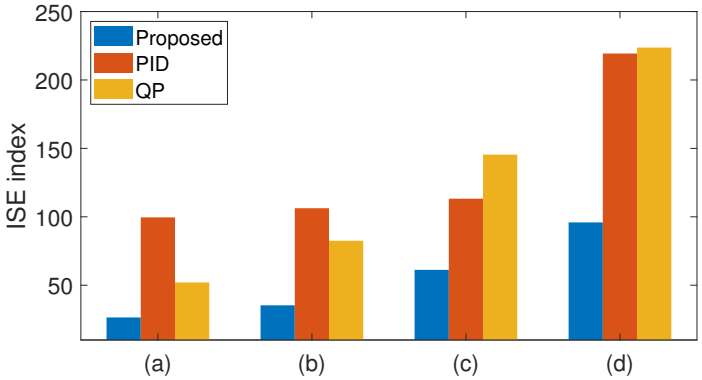


Figure 2.9: Summary of the ISE performance comparisons. (a) Slow tracking without uncertainty. (b) Fast tracking without uncertainty. (c) Slow tracking with uncertainty. (d) Fast tracking with uncertainty.

2.5.2 Effectiveness of proposed controller

To reveal the insights of the EID-based controller (2.17) for tracking purposes, we point out the specific role of each involved control component, i.e., feedback control u_{fb}, feedforward control u_{ff}, and disturbance-compensation based control u_{dc}. To this end, we reconsider Scenario 2 with a fast trajectory tracking task, decomposed into three phases:

- Phase 1: From 0 [s] to 13 [s] with only feedback control, i.e., controller (2.17) with $u_{ff} = 0$ and $u_{dc} = 0$.
- Phase 2: From 13 [s] to 23 [s] with feedback and feedforward control, i.e., controller (2.17) with $u_{dc} = 0$.
- Phase 3: From 23 [s] to 33 [s] with the complete EID-based controller (2.17).

The tracking control result along the $y-$axis direction is shown in Figure 2.10. During Phase 1, the tracking is performed with significant errors in amplitude and in phase. Integrating the feedforward control action in Phase 2 can only improve the phase error since the amplitude error is still observed due to the modeling uncertainty. Using the EID-based control law (2.17) in Phase 3, the amplitude error can be also significantly reduced with the error-compensation term u_{dc}. Hence, the proposed method allows to achieve an effective fast (and slow) tracking control for soft robots despite the modeling errors.

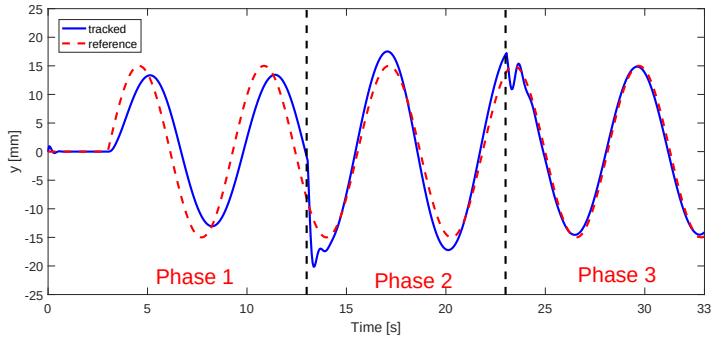


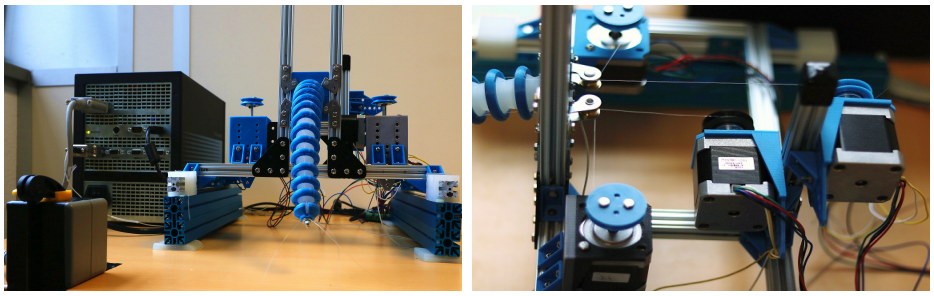
Figure 2.10: Three-phase fast trajectory tracking along the $y-$axis in Scenario 2.

2.5.3 Experimental Validations

Hereafter, we present the tracking control results, experimentally obtained with the Trunk robot shown in Figure 2.11.

Hardware Setup

The silicone Trunk robot is composed of 14 segments to make it highly deformable. This soft robot is driven by four stepper motors via cables mounted on the robot body to guarantee the accessibility of each direction in the workspace. The position



(a) (b)

Figure 2.11: Trunk robot. (a) Robot platform, (b) Stepper motors.

of the end-effector, e.g., system output, is measured by a Polhemus Liberty magnetic motion tracking sensor. The weight of the Trunk robot is 40 [g] and its length is 195 [mm] in the initial position. The FEM model of this robot has 1944 nodes, leading to 3324 state variables. After performing the POD model reduction, a reduced four-order dynamical Trunk robot model can be obtained for control design, while around 90% percent of singular values can be kept. Since the workspace of the Trunk robot end effector is a surface, then it is sufficient to determine the position of the end-effector with a 2D reference trajectory. For tracking control purposes, we consider the spiral trajectory

$$x(t) = 0.2t\cos(\omega t), \quad y(t) = 0.2t\sin(\omega t), \tag{2.44}$$

with $\omega = 1$ [rad/s], corresponding to a fast trajectory tracking. Note that the linear speed of the spiral reference is continuously increasing according to the radius, making the tracking task more challenging.

Tracking Control Performance

For this trajectory tracking, we examine the experimental results obtained with two dynamic control methods: Jacobian-based PID control and EID-based control, while the QP-based method is not applied due to the platform limitation. Figure 2.12 depicts the evolutions of the end-effector position tracking in the $xy-$plane, performed by both considered controllers. We can see clearly that compared to Jacobian-based PID control, the proposed EID-based control method provides a significant improvement in terms of tracking performance. The tracking performance along $x-$axis and $y-$axis directions, and the corresponding force control inputs obtained with the spiral reference trajectory (2.44) are given in Figure 2.13. Observe in

Figs. 2.13(c) and (d) that due to the effects of the feedforward action, the proposed EID-based controller provides a faster response with respect to the Jacobian-based PID controller, which allows for a better dynamic tracking performance as shown in Figs. 2.13(a) and (b). We can also see in Figs. 2.13(c) and (d) that the control inputs of both controllers start at the same values corresponding to the robot equilibrium point. Moreover, there is only a small amplitude difference concerning the $x-$axis force control inputs. However, along the $y-$axis, the proposed EID-based controller provides a larger control action to better compensate the time-varying disturbance due to the gravity effect, which improves the control precision performance as indicated in Figure 2.12. For quantitative comparison purposes, the ISE indices, defined in (2.43), are computed for both EID-based control and Jacobian-based PID control over a time duration $\Delta t = 22$ [s] as 153.3 [mm^{2}s] and 778.9 [mm^{2}s], respectively. For this experimental test, we can note a tracking performance improvement ISE index from 7.78 to 0.53. Videos of these experiments can be found at the address: https://bit.ly/2VVwtLn.

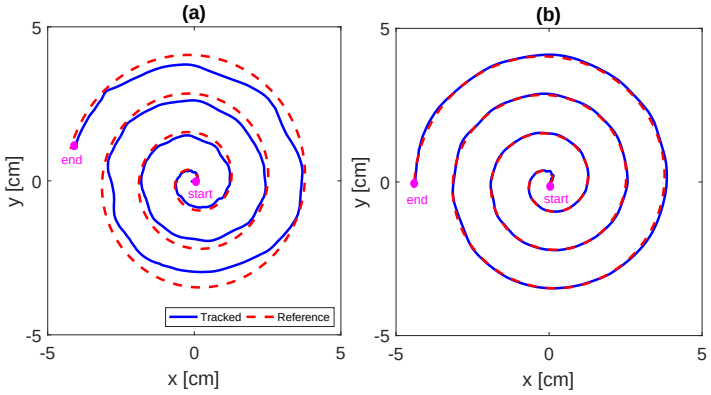


Figure 2.12: Experimental results of spiral trajectory tracking control. (a) Jacobian-PID control. (b) Proposed EID-based control.

2.6 Concluding Remarks

A dynamic FEM model-based framework has been proposed for the tracking control of soft robots. For the control design, the large-scale FEM robot model is effectively reduced with a POD model reduction method. Using an EID approach, we propose an observer-based tracking control structure including three components, i.e., feedback control, feedforward control, error-compensation control. This control

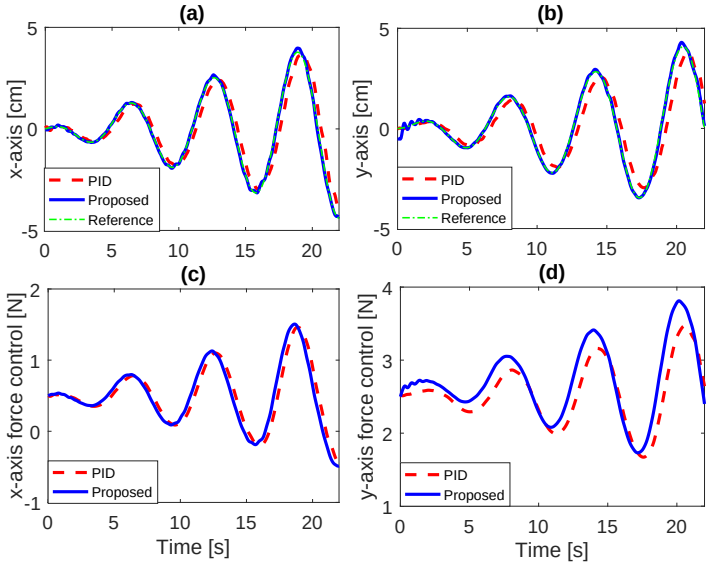


Figure 2.13: Comparison of spiral trajectory tracking control. (a) Tracking performance along $x-$axis. (b) Tracking performance along $y-$axis. (c) Force control input in $x-$axis direction. (d) Force control input in $y-$axis direction.

structure allows for an effective tracking control performance despite the presence of modeling uncertainty and unknown disturbances. To improve the closed-loop responses, the concept of LMI regions is exploited together with an ℓ_∞ control design via Lyapunov stability theory. The effectiveness of the proposed control method has been first demonstrated with high-fidelity SOFA simulations performed on a Diamond soft robot. Then, experimental validations have been also carried out with a Trunk robot. Under the considered experimental conditions, the proposed control method shows a clear tracking performance improvement compared to the existing Jacobian-PID control method, i.e., about 80% of improvement in terms of ISE index.

The work done in this chapter makes possible an extension to linear systems with varying parameters (LPV). It allows to take into account the nonlinearities and modeling uncertainties caused by the large deformations of soft robots. The fact that model reduction preserves the initial structure of the system is crucial for determining controllers. The next chapter proposes to extend the results to the synthesis of a nonlinear controller valid in the entire workspace of the soft robot.

Appendix A: Proof of Theorem 1

By Lemma 1, we can show that LMI conditions (2.34), (2.35) and (2.36) (respectively (2.37), (2.38) and (2.39)) guarantee that the eigenvalues of the dynamic matrix $\bar{A}_c$ of the tracking error system (2.27) (respectively $\bar{A}_o$ of the estimation error system (2.24)) are confined in the LMI region $\mathscr{D}_c(r_c, \theta_c, \alpha_c)$ (respectively $\mathscr{D}_o(r_o, \theta_o, \alpha_o)$).

Pre- and postmultiplying inequality (2.34) with $P = X^{-1}$ while considering (2.41), it follows that

$$\mathrm{He}\left(P\bar{A}_c + \alpha_c P\right) \prec 0. \tag{2.45}$$

Let us denote

$$\Pi_c = \mathrm{He}\left(P\bar{A}_c + \alpha_c P\right), \quad \Pi_o = \mathrm{He}\begin{bmatrix} \mathscr{A}_o + \alpha_c Q & QB_{od} \\ 0 & -\alpha_c \nu I \end{bmatrix}.$$

Note that $\Pi_o \prec 0$ and $\Pi_c \prec 0$ due to conditions (2.33) and (2.45), respectively. Since Π_c only depends on the control gain K_c and matrix P, and Π_o only depends on the observer gain L_o and matrix Q, there always exists a positive scalar $\tau > 0$, sufficiently small, such that [68]

$$\Pi_o \preceq \tau \Phi^\top \Pi_c^{-1} \Phi, \tag{2.46}$$

with $\Phi = \begin{bmatrix} PB_o & PB_{cd} \end{bmatrix}$. By Schur complement lemma [69], we can prove that condition (2.46) is equivalent to

$$\begin{bmatrix} \tau\Pi_c & \tau\Phi \\ \star & \Pi_o \end{bmatrix} \preceq 0. \tag{2.47}$$

From the definitions of Π_c, Π_o, and Φ, condition (2.47) can be rewritten as

$$\mathrm{He}\begin{bmatrix} \tau P(\bar{A}_c + \alpha_c I) & \tau PB_o & \tau PB_{cd} \\ 0 & Q(\bar{A}_o + \alpha_c I) & QB_{od} \\ 0 & 0 & -\alpha_c \nu I \end{bmatrix} \preceq 0. \tag{2.48}$$

To study the stability of the extended closed-loop system (2.28), we consider a Lyapunov function candidate as

$$V(\tilde{x}) = \tilde{x}^\top \mathrm{diag}(\tau P, Q)\tilde{x}. \tag{2.49}$$

Now, pre- and postmultiplying (2.48) with $\begin{bmatrix} \tilde{x}^\top & d_{\dot{q}}^\top \end{bmatrix}^\top$ and its transpose, the following condition can be obtained after some algebraic manipulations:

$$\dot{V}(\tilde{x}) \leq -2\alpha_c (V(\tilde{x}) - \nu d_{\dot{q}}^\top d_{\dot{q}}), \tag{2.50}$$

where $\dot{V}(\tilde{x})$ is the time derivative of the Lyapunov function (2.49) along the trajectory of the closed-loop system (2.28). From the relation of vector-norms, inequality (2.50) implies that

$$\dot{V}(\tilde{x}) \leq -2\alpha_c \left(V(\tilde{x}) - \nu \|d_{\dot{q}}\|_\infty^2\right). \tag{2.51}$$

Applying the comparison lemma [44, Lemma 3.4] to (2.51), it follows that

$$V(\tilde{x}(t)) \leq e^{-2\alpha_c t} V(\tilde{x}(0)) + \nu \|d_{\dot{q}}\|_\infty^2. \tag{2.52}$$

Note that

$$\alpha_1 \|\tilde{x}\|^2 \leq V(\tilde{x}) \leq \alpha_2 \|\tilde{x}\|^2, \tag{2.53}$$

with $\alpha_1 = \lambda_{\min}(\mathrm{diag}(\tau P, Q))$ and $\alpha_2 = \lambda_{\max}(\mathrm{diag}(\tau P, Q))$. It follows from (2.52) and (2.53) that

$$\|\tilde{x}\| \leq \sqrt{\frac{\alpha_2}{\alpha_1}} e^{-\alpha_c t} \|\tilde{x}(0)\| + \sqrt{\frac{\nu}{\alpha_1}} \|d_{\dot{q}}\|_\infty, \tag{2.54}$$

which guarantees Property (P1), i.e., the input-to-state stability of system (2.28) with respect to any disturbance $d_{\dot{q}} \in \mathscr{B}_\infty$.

Let us consider the definition of the performance output z in (2.29). Pre- and postmultiplying inequality (2.40) with $\mathrm{diag}(P, I, I)$, it follows that

$$\begin{bmatrix} \mathrm{diag}(P, Q) & F^\top \\ \star & \mu I \end{bmatrix} \succeq 0. \tag{2.55}$$

Applying Schur complement lemma and congruence transformation [69] to inequality (2.55), it follows that

$$\mu \mathrm{diag}(\tau P, Q) - F^\top F \succeq 0, \tag{2.56}$$

with $\tau > 0$. Pre- and postmultiplying condition (2.56) with $\tilde{x}^\top$ and its transpose yields

$$\|z\|^2 \leq \mu V(\tilde{x}). \tag{2.57}$$

From (2.52) and (2.57), we can deduce that

$$\|z(t)\| \leq \sqrt{\mu V(\tilde{x}(0))} e^{-\alpha_c t} + \sqrt{\nu\mu} \|d_{\dot{q}}\|_\infty. \tag{2.58}$$

For any initial condition $\tilde{x}(0)$ and any bounded signal $d_{\dot{q}}$, it follows from (2.58) that

$$\lim_{t \to \infty} \sup \|z(t)\| \leq \gamma \|d_{\dot{q}}\|_\infty, \tag{2.59}$$

which guarantees Property (P2). This concludes the proof.

Chapter 3

Data-Driven LPV Modeling and Control Design

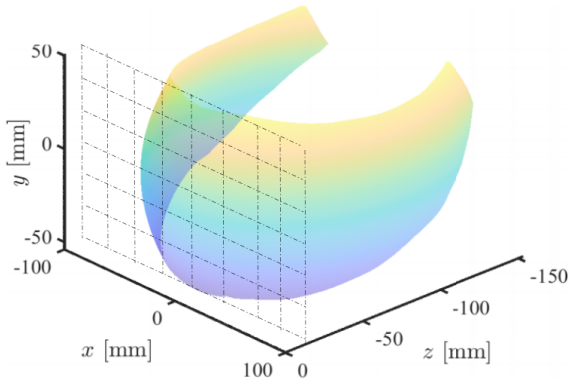


Figure 3.1: Workspace of Trunk robot

Figure 3 shows the shape of the Trunk robot workspace, the linearized model is only applicable to the linear plane as shown with the mesh. We can see that the deformations involve a nonlinear surface making the linear method of the previous chapter impossible to apply on the whole surface. In order to perform a control in the whole workspace it is necessary to perform a synthesis of the control from a nonlinear model of the robot. Since an analytical control oriented model is intractable, we will construct a LPV model from linear models obtained from FEM. In this chapter, a stepwise procedure of LPV modeling and control design method is proposed including extensive experiment validations to show their effectiveness.

3.1 Introduction

As discussed in chapter 1, the main difficulties in the control of soft robots are the models accuracies including the strong nonlinearities. We have proposed a framework from linearization to model reduction in Chapter 2 to obtain control-oriented linear models. However, due to the strong nonlinearity of soft robots, linear models are only effective in a reduced range. Since the FEM is an iteratively updated model, we cannot directly obtain a nonlinear model from it, nevertheless, a group of linear models can be acquired for various equilibrium points. The problem we face is how to use these linear models to obtain a global nonlinear control.

The primary goal of this chapter is to propose a generic solution for this critical issue in soft robotics via a model-based nonlinear dynamic control scheme. The proposed nonlinear control framework for soft robots is sequentially performed in two steps: i) linear parameter varying (LPV) modeling, and ii) tracking LPV control design.

For LPV modeling, we first perform linearization at several equilibrium points, densely selected to cover the whole workspace of the soft robot. From the resulting large-scale models, a proper orthogonal decomposition (POD) algorithm is used to generate a set of linearized reduced-order models, which represent the local behaviors of the soft robot corresponding to the selected equilibrium points. With a unified POD projector for order reduction, the coherence between the linearized models is naturally ensured, which is essential for interpolation-based LPV modeling [70], [71]. For LPV interpolation, due to the large number of linearized models and their correlation with each other, we propose a decorrelation algorithm to limit the numerical complexity of the LPV model, i.e., reduce the number of local linear submodels, while guaranteeing a high-quality LPV modeling. To this end, the algorithm builds the LPV model of the soft robot by interpolating a subset of selected linearized models with radial basis function (RBF) networks.

Concerning the LPV tracking control scheme, it is generalized from the equivalent input disturbance (EID) control concept [57] used in the previous chapter, whose core components are feedforward control, disturbance-estimator control and feedback control. A generalized proportional integral LPV observer is designed to provide not only the estimate of reduced-order states for feedback control but also the disturbance-estimator control action to compensate matched disturbances, e.g., modeling errors and external loads. The closed-loop stability of the soft robot system is guaranteed by the feedback control, whose design is reformulated as a convex optimization problem under linear matrix inequality (LMI) constraints. Using Lyapunov stability theory, an $\ell_\infty-$gain performance is incorporated in the feedback con-

trol design to improve the tracking performance under the presence of unknown and unmatched disturbances. The main contributions of this chapter can be summarized as follows.

- A constructive LPV modeling method, including FEM-based modeling and discretization, reduced-order reduction and iterative RBF-based interpolation, which is relevant to represent and to control the nonlinear dynamics of soft robots within the whole workspace.
- An ℓ_∞ LPV control framework with formal proofs of closed-loop stability, which requires only the output information to achieve an effective tracking performance under unknown disturbances.
- Experimental results and suitable comparative studies are conducted with a soft Trunk robot to fully validate the proposed LPV control framework in terms of nonlinear modeling and tracking performance.

Due to its generic feature, the proposed LPV control methodology can be applied to a large variability of elastic soft robots.

3.2 LPV Modeling of Soft Robots

This section presents a new LPV modeling method for large-scale soft robots based on their FEM models and the RBF interpolation method. The linearization and the model reduction methods follow the one presented in the previous chapter. Thereafter, they are developed in a discrete framework for convenience and controller design implementation.

3.2.1 FEM-Based Reduced-Order Models

Linearized State-Space Representation

Here we recall the linearized constitutive equation of the FEM robot model (2.4) around the equilibrium point $(\boldsymbol{q}_0, \boldsymbol{v}_0, \boldsymbol{u}_0)$ given by

$$M\dot{\boldsymbol{v}}(t) = -K\boldsymbol{q}(t) - D\boldsymbol{v}(t) + H\boldsymbol{u}(t). \tag{3.1}$$

Here $\boldsymbol{v}(t)$ is the generalized velocity given with $\boldsymbol{v}(t) = \frac{\partial \boldsymbol{q}(t)}{\partial t}$. Due to the research progress in the discrete control system, discrete system models give us more possibilities to model systems and design less conservative controllers and it is also easier

for the real-time implementation. To this end, we will perform the control design in the discrete-time domain. For the large-scale model (3.1), to improve the numerical efficiency and stability, the implicit Euler method can be used [72]. Considering a sampling time step h and a constant control input $\boldsymbol{u}$ during the sampling interval, the implicit Euler-discretization of model (3.1) is given by

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + h\boldsymbol{v}_{k+1} \tag{3.2}$$
$$M\boldsymbol{v}_{k+1} = M\boldsymbol{v}_k + h(-K\boldsymbol{q}_{k+1} - D\boldsymbol{v}_{k+1} + H\boldsymbol{u}_k). \tag{3.3}$$

It follows from (3.2) and (3.3) that

$$S\boldsymbol{v}_{k+1} = -hK\boldsymbol{q}_k + M\boldsymbol{v}_k + hH\boldsymbol{u}_k \tag{3.4}$$

with $S = M + hD + h^2K$. From (3.2) and (3.4), the discrete-time robot model can be reformulated as

$$E\begin{bmatrix}\boldsymbol{q}_{k+1}\\ \boldsymbol{v}_{k+1}\end{bmatrix} = \begin{bmatrix} I & 0\\ -hS^{-1}K & S^{-1}M\end{bmatrix}\begin{bmatrix}\boldsymbol{q}_k\\ \boldsymbol{v}_k\end{bmatrix} + \begin{bmatrix}0\\ hS^{-1}H\end{bmatrix}\boldsymbol{u}_k \tag{3.5}$$

with $E = \begin{bmatrix} I & -hI\\ 0 & I\end{bmatrix}$. Left-multiplying (3.5) with E^{-1}, the following state-space robot model can be obtained:

$$\begin{bmatrix}\boldsymbol{q}_{k+1}\\ \boldsymbol{v}_{k+1}\end{bmatrix} = \begin{bmatrix} I - h^2S^{-1}K & hI - h^2S^{-1}(D+hK)\\ -hS^{-1}K & I - hS^{-1}(D+hK)\end{bmatrix}\begin{bmatrix}\boldsymbol{q}_k\\ \boldsymbol{v}_k\end{bmatrix} + \begin{bmatrix}h^2S^{-1}H\\ hS^{-1}H\end{bmatrix}\boldsymbol{u}_k. \tag{3.6}$$

With a small sampling time step h, the impact of the terms related to h^2, i.e., $h^2S^{-1}K$, $h^2S^{-1}H$ and $h^2S^{-1}(D+hK)$, on the modeling quality are supposed to be small enough to be removed. After neglecting these terms in (3.6), we obtain the simplified model

$$\boldsymbol{x}_{L,k+1} = A_L\boldsymbol{x}_{L,k} + B_L\boldsymbol{u}_k \tag{3.7}$$

where $\boldsymbol{x}_{L,k} = \begin{bmatrix}\boldsymbol{q}_k^\top & \boldsymbol{v}_k^\top\end{bmatrix}^\top \in \mathbb{R}^{2m}$ is the state vector, and the matrices $A_L \in \mathbb{R}^{2m\times 2m}$ and $B_L \in \mathbb{R}^{2m\times p}$ are defined as

$$A_L = \begin{bmatrix} I & hI\\ -hS^{-1}K & I - hS^{-1}D\end{bmatrix}, \quad B_L = \begin{bmatrix}0\\ hS^{-1}H\end{bmatrix}. \tag{3.8}$$

The system output $\boldsymbol{y}_k \in \mathbb{R}^q$ represents the coordinates of the robot end-effectors, defined as

$$\boldsymbol{y}_k = C_L \boldsymbol{x}_{L,k} \tag{3.9}$$

where the non-zero elements of the output matrix C_L correspond to the states of the end-effectors.

POD-Based Model Order Reduction

Same as equation (2.7) in chapter 2, the projector T of the POD method is also constructed from SVD of snapshot matrix according to a desired number r of singular values. We continue using the notation $T = U_r$, where $T \in R^{r\times m}$ contains the first r columns, with $r \ll m$, of matrix U. Then, the reduced state vector is obtained with [12]

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{v}_r \end{bmatrix} = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{bmatrix} = T_r \boldsymbol{x}_L. \tag{3.10}$$

Applying the projector (3.10) to the large-scale linearized system (3.7)–(3.9), we obtain the reduced-order model of the form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A_r \boldsymbol{x}_k + B_r \boldsymbol{u}_k + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C \boldsymbol{x}_k \end{aligned} \tag{3.11}$$

where the disturbance $\boldsymbol{w}_k$ represents the model order reduction error, and

$$\begin{aligned} A_r &= T_r A_L T_r^\top = \begin{bmatrix} I & hI \\ -hTS^{-1}KT^\top & I - hTS^{-1}DT^\top \end{bmatrix} \\ B_r &= T_r B_L = \begin{bmatrix} 0 \\ hTS^{-1}H \end{bmatrix}, \quad C = C_L T_r^\top. \end{aligned} \tag{3.12}$$

3.2.2 RBF-Based LPV Modeling

Due to the large non-linearity of soft robots, a control design based on a linearized robot model (3.11) may not be effective for different robot configurations. For instance, Figure 3.2 illustrates the nonlinear behavior of the Trunk robot during a bending process. We can see that increasing the actuation force can lead to different directions of displacement along the $x-$axis. Indeed, the $x-$axis displacement first increases till a certain singular position, then it starts to decrease if the force continues to increase. Moreover, the variation of the curve slope shows that the stiffness of the Trunk robot increases with respect to the increment of actuation forces.

To deal with such position-dependent nonlinear dynamics of a deformable robot, we propose an LPV modeling method based on a family of local reduced-order models as in (3.11).

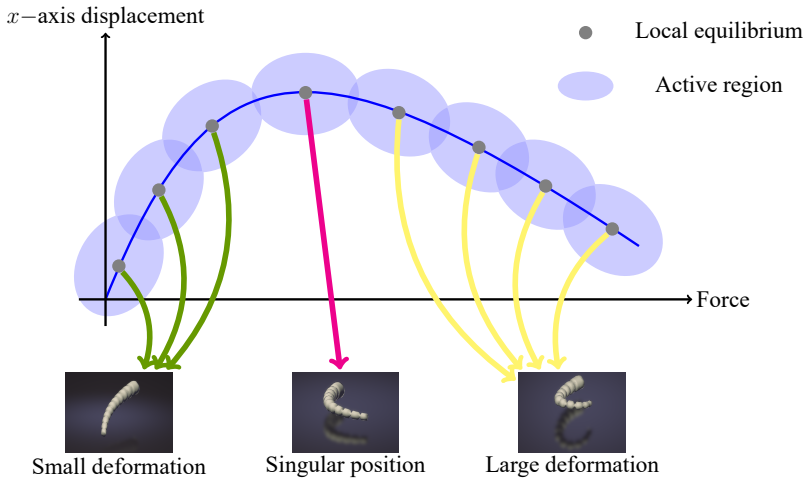


Figure 3.2: Illustration of nonlinear behaviors of the Trunk robot during bending.

Generating LPV Local Linear Models

To capture the complex dynamics of a soft robot, we select an appropriate number of equilibrium points as illustrated in Figure 3.3 to cover as large as possible its whole workspace. For each equilibrium point $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$, with $\boldsymbol{x}_i^* = \begin{bmatrix}\boldsymbol{q}_{ri}^{*\top} & \boldsymbol{v}_{ri}^{*\top}\end{bmatrix}^\top$, we first obtain the linearized FEM model (3.1) and the implicit Euler-discretization as in (3.2)–(3.3). Then, the POD-based order reduction (3.10)–(3.10) is applied to obtain the reduced-order model as in (3.11):

$$\begin{aligned}\delta\boldsymbol{x}_{i,k+1} &= A_{ri}\delta\boldsymbol{x}_{i,k} + B_{ri}\delta\boldsymbol{u}_{i,k} + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C(\delta\boldsymbol{x}_{i,k} + \boldsymbol{x}_i^*)\end{aligned} \tag{3.13}$$

where $\delta\boldsymbol{x}_{i,k} = \boldsymbol{x}_k - \boldsymbol{x}_i^* \in \mathbb{R}^n$, with $n = 2r$, and $\delta\boldsymbol{u}_{i,k} = \boldsymbol{u}_k - \boldsymbol{u}_i^* \in \mathbb{R}^p$. Note that the state $\delta\boldsymbol{x}_{i,k}$ of the ith linearized model (3.13) is defined as a relative distance between the global coordinate $\boldsymbol{x}_k$ and its local equilibrium $\boldsymbol{x}_i^*$. For simplicity of notation, we use the same disturbance vector $\boldsymbol{w}_k$ to represent the reduction errors in different reduced-order models. Similar to (3.12), the state-space matrices $A_{ri} \in \mathbb{R}^{n\times n}$ and

$B_{ri} \in \mathbb{R}^{n\times m}$ are given by

$$A_{ri} = \begin{bmatrix} I & hI \\ \mathcal{K}_{ri} & \mathcal{D}_{ri} \end{bmatrix}, \quad B_{ri} = \begin{bmatrix} 0 \\ \mathcal{H}_{ri} \end{bmatrix} \tag{3.14}$$

with

$$\begin{aligned} \mathcal{K}_{ri} &= -hTS_i^{-1}K_iT^\top, \\ \mathcal{D}_{ri} &= I - hTS_i^{-1}D_iT^\top, \\ \mathcal{H}_{ri} &= hTS_i^{-1}H_i. \end{aligned} \tag{3.15}$$

The matrices S_i, K_i, D_i and H_i in (3.15) corresponding to the equilibrium point $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$ are obtained in a similar way as S, K, D and H in (3.8), respectively. It is important to note that since the same projector T_r in (3.10) is applied to all linearized submodels, the corresponding reduced models have the same order. Note also that the state $\delta\boldsymbol{x}_{i,k}$ of each linearized model (3.13) only has a local meaning, i.e., relative distance between $\boldsymbol{x}_k$ and $\boldsymbol{x}_i^*$. Then, interpolating these linearized local models has no meaning. To deal with this misfit, we reformulate the local model (3.13) in function of the global state $\boldsymbol{x}_k$ of the robot as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A_{ri}\boldsymbol{x}_k + B_{ri}\boldsymbol{u}_k + \boldsymbol{\zeta}_{ri} + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C\boldsymbol{x}_k. \end{aligned} \tag{3.16}$$

The constant affine term $\boldsymbol{\zeta}_{ri}$ of the ith local model can be defined from the equilibrium point $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$ as

$$\boldsymbol{\zeta}_{ri} = (I - A_{ri})\boldsymbol{x}_i^* - B_{ri}\boldsymbol{u}_i^*. \tag{3.17}$$

At the equilibrium, we have $\boldsymbol{v}_{ri}^* = 0$. Then, substituting (3.14) into (3.17), the term $\boldsymbol{\zeta}_{ri}$ can be expressed as

$$\boldsymbol{\zeta}_{ri} = \begin{bmatrix} 0 \\ -\mathcal{K}_{ri}\boldsymbol{q}_{ri}^* - \mathcal{H}_{ri}\boldsymbol{u}_i^* \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{\zeta}_{ri}^* \end{bmatrix} \tag{3.18}$$

with $\boldsymbol{\zeta}_{ri}^* \in \mathbb{R}^r$.

RBF-Based Interpolation

To represent the position-dependent nonlinear dynamics of soft robots, we select the end-effector position as the scheduling vector $\boldsymbol{\theta}$ for LPV modeling, i.e., $\boldsymbol{\theta} = \boldsymbol{y}$. RBF networks allow to construct accurate interpolation of unstructured data,

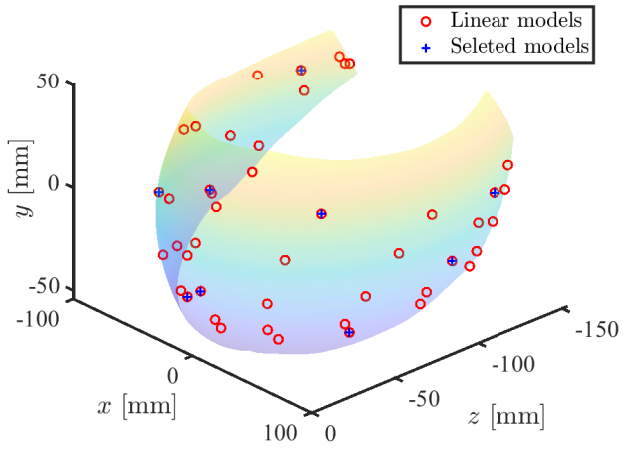


Figure 3.3: Workspace of the soft Trunk robot studied in Section 3.5 and selected linearized models for its RBF-based LPV modeling.

possibly in high-dimensional spaces, with a small number of radial basis functions, which is particularly suitable for nonlinear modeling of soft robots. The output $\psi(\boldsymbol{\theta})$ of a normalized RBF network is defined as [73]:

$$\psi(\boldsymbol{\theta}) = \sum_{i=1}^{N} \boldsymbol{a}_i \eta_i(\boldsymbol{\theta}) \tag{3.19}$$

where the scheduling variable $\boldsymbol{\theta}$ is the input vector of the RBF network, N is the number of RBFs, and $\boldsymbol{a}_i$ is the output parameter of the ith RBF. The normalized radial basis function $\eta_i(\boldsymbol{\theta})$ is defined as

$$\eta_i(\boldsymbol{\theta}) = \frac{\varphi_i\left(\|\boldsymbol{\theta} - \boldsymbol{c}_i\|\right)}{\sum_{j=1}^{N} \varphi_j\left(\|\boldsymbol{\theta} - \boldsymbol{c}_j\|\right)}, \quad i \in \mathcal{I}_N \tag{3.20}$$

where $\boldsymbol{c}_i$ is the center vector for the ith RBF, which represents the equilibrium point $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$ corresponding to the ith linearized model (3.16). As usual in practice, we select the Gaussian function as the radial basis function $\varphi_i\left(\|\boldsymbol{\theta} - \boldsymbol{c}_i\|\right)$ in (3.20)

$$\varphi_i\left(\|\boldsymbol{\theta} - \boldsymbol{c}_i\|\right) = e^{-(\varepsilon\|\boldsymbol{\theta} - \boldsymbol{c}_i\|)^2} \tag{3.21}$$

where the parameter ε is the width of the receptive field.

Remark 6. The Gaussian radial basis function (3.21) is used as a similarity measure, i.e., the similarity reaches the maximum at the center $\boldsymbol{c}_i$ and decreases when the

scheduling variable $\boldsymbol{\theta}$ is far away from $\boldsymbol{c}_i$. Then, the key idea of the proposed LPV modeling is to represent the dynamics of a soft robot at any given point in the workspace by interpolating the nearest linearized models.

Remark 7. RBF networks are universal approximators, which can approximate any continuous function on a compact set with an arbitrary precision [74]. However, the radial basis functions of conventional RBF networks are constructed according to each data sample, i.e., each data sample requires its own RBF [75]. Then, to represent accurately the high nonlinear dynamics, a large number of RBFs N, i.e., a large number of linearized models, may be required, which induces difficulties for the control design and real-time implementation of soft robots. To avoid this major issue, based on the idea of the CPR algorithm [76], we propose an interpolator being able to effectively represent a large number of local linearized models using as less as possible number of RBFs N, as described below.

RBF networks are typically trained from pairs of input and target values. To start the training of the proposed RBF network, we first collect a dataset of linearized models (3.16) of M equilibrium points $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$, corresponding to the input vectors $\boldsymbol{\theta}_i^* = \boldsymbol{y}_i^*$, for $i \in \mathcal{I}_M$. Then, for each collected data sample, we reshape the data into a column vector $Y_{(i)}$, for $i \in \mathcal{I}_M$, containing the parameters of $\mathcal{K}_{ri}$, $\mathcal{D}_{ri}$ and $\mathcal{H}_{ri}$ involved in the matrices A_{ri} and B_{ri} in (3.14), and in the affine term $\boldsymbol{\zeta}_{ri}^*$ defined in (3.17). Note that since the linearized robot models are highly correlated, a sparse representation of these submodels can be performed. To this end, similar to the CPR algorithm [76], we iteratively define the centers of the proposed RBF network. For each step, a new RBF is added in the center position corresponding to the largest squared error, until a given number of functions or a desired fitting performance is reached. The iterative training procedure is summarized in Algorithm 1 and illustrated in Figure 3.4. The network weights are obtained from the Orthogonal Least Square Learning algorithm with respect to all training data as described in [77].

After the training, the dataset of M linearized models can be interpolated by an $N-$ element RBF network, with $N \leq M$. Hence, the resulting interpolated LPV model has N local linear submodels, defined as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A(\eta)\boldsymbol{x}_k + B(\eta)\boldsymbol{u}_k + \boldsymbol{\zeta}(\eta) + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C\boldsymbol{x}_k \end{aligned} \tag{3.22}$$

with

$$\begin{bmatrix} A(\eta) & B(\eta) & \boldsymbol{\zeta}(\eta) \end{bmatrix} = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) \begin{bmatrix} A_i & B_i & \boldsymbol{\zeta}_i \end{bmatrix}. \tag{3.23}$$

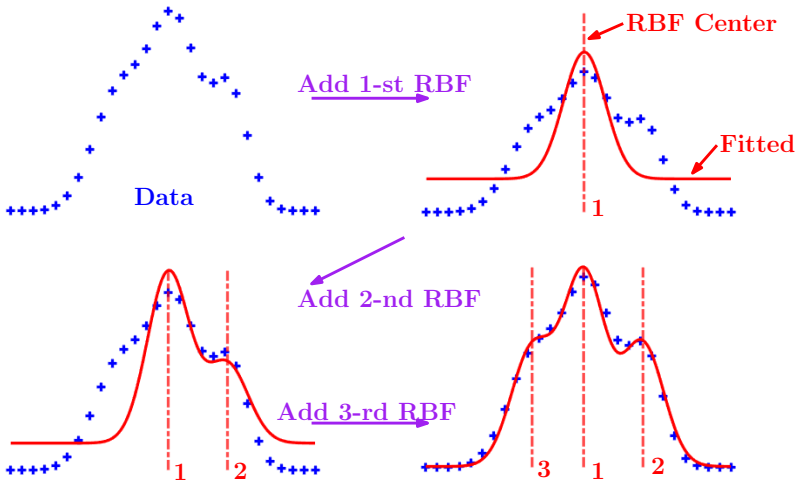


Figure 3.4: Illustration of the iterative RBF training procedure in Algorithm 1. Here, the sample data are fitted using three radial basis functions.

The radial basis functions $\eta_i(\boldsymbol{\theta}_k)$, for $i \in \mathcal{I}_N$, are used as the membership functions of the LPV model (3.22), i.e., to weight local linear submodels, which satisfy the convex sum property

$$0 \leq \eta_i(\boldsymbol{\theta}_k) \leq 1, \quad \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) = 1, \quad i \in \mathcal{I}_N. \tag{3.24}$$

Let Ω be the set of membership functions satisfying (3.24), i.e.,

$$\eta = \begin{bmatrix} \eta_1(\boldsymbol{\theta}_k) & \dots & \eta_N(\boldsymbol{\theta}_k) \end{bmatrix}^\top \in \Omega,$$

for $\forall k \in \mathbb{N}$. We also denote $\eta_+ = \begin{bmatrix} \eta_1(\boldsymbol{\theta}_{k+1}), \eta_2(\boldsymbol{\theta}_{k+1}), \dots, \eta_N(\boldsymbol{\theta}_{k+1}) \end{bmatrix}^\top \in \Omega$. With the proposed LPV modeling method, the parameter structures of A_i, B_i and $\boldsymbol{\zeta}_i$ in (3.23) are preserved as the same as those of A_{ri}, B_{ri} and $\boldsymbol{\zeta}_{ri}$ in (3.14) and (3.18), which are given by

$$\begin{aligned} A_i &= \begin{bmatrix} I & hI \\ \mathcal{K}_i & \mathcal{D}_i \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 \\ \mathcal{H}_i \end{bmatrix} \\ \boldsymbol{\zeta}_i &= \begin{bmatrix} 0 \\ -\mathcal{K}_i \boldsymbol{q}_{ri}^* - \mathcal{H}_i \boldsymbol{u}_i^* \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{\zeta}_i^* \end{bmatrix}. \end{aligned} \tag{3.25}$$

The parameter matrices $\mathcal{K}_i$, $\mathcal{D}_i$, $\mathcal{H}_i$, and the affine term $\boldsymbol{\zeta}_i^*$ in (3.25) are constructed from the reshaped data in the output column vector $\boldsymbol{a}_i$, for $i \in \mathcal{I}_N$, defined in (3.19).

The validation of the proposed LPV modeling with a soft Trunk robot is reported in Section 3.5.

Algorithm 1 Iterative Training of RBF Network

Inputs:
Data set $\mathscr{D} = \left\{ \left(\theta_1^*, Y_{(1)}\right), \ldots, \left(\theta_M^*, Y_{(M)}\right) \right\}$
Outputs:
Weights $\boldsymbol{a}_i$ and centers $\boldsymbol{c}_i$ of RBFs, for $i \in \mathcal{I}_M$
Required parameters:
Maximum number N of the RBF network
Width of the receptive field ε of the RBF network
Desired interpolation error σ
Initialization:
- Set the estimates of all $Y_{(i)} = 0$ as $\hat{Y}_{(i)} = 0$, for $i \in \mathcal{I}_M$
- Initialize an empty set $\mathscr{S}$ of weights $\boldsymbol{a}_i$ and centers $\boldsymbol{c}_i$ for the RBF network

Begin
for $i = 1$ **to** N **do**
  for $j = 1$ **to** i **do**
    - Compute the activation vector of RBFs
      $$\Phi_j = [\eta_j(\theta_1^*), \eta_j(\theta_2^*), \ldots, \eta_j(\theta_M^*)]^\top$$
    - Build the activation matrix $\Psi = [\Phi_1, \ldots, \Phi_i]^\top$
    - Build the coefficient matrix $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_i]$
  end for
  - Compute the estimation output $\hat{Y} = \mathbf{A}\Psi$
  - Find the index k, for $k \in \mathcal{I}_M$, corresponding to the largest prediction error $\|\hat{Y}_{(k)} - Y_{(k)}\|^2$
  - Add a new RBF with the center $\boldsymbol{c}_k = \theta_k^*$ into the set $\mathscr{S}$
  - Update the activation matrix Ψ with the new RBF
  - Recompute the coefficients $\boldsymbol{a}_k$ from the least square solution of $Y = \mathbf{A}\Psi$ with respect to $\mathbf{A}$
  if $\sum_{k=1}^{N} \|\hat{Y}_{(k)} - Y_{(k)}\|^2 < \sigma$ **then**
    `break`
  end if
end for
End

Remark 8. Existing data-driven LPV modeling methods usually include independent model identification and LPV model construction processes. From system data, local linear submodels are directly identified, which are generally independent from each other, also called non-coherent [71]. Interpolating non-coherent submodels requires some specific canonical state-space representations, which may not only lead to a loss of the mechanical model structure of soft robots but also alter the system behaviors [78]. For the proposed LPV modeling, the coherence between local linearized submodels can be naturally guaranteed by the unified POD projector T in (3.10). Moreover, since the mechanical structure is preserved, the upper-half block-elements of A_i, B_i and $\boldsymbol{\zeta}_i$ given in (3.25) are constant. Hence, the number of parameters to be interpolated with the proposed LPV modeling is reduced by 50%.

3.2.3 Modeling Error Analysis

Soft-body robots have much larger modeling and manufacturing errors than rigid-body robots. However, since the structure of the system is preserved during the modeling process, we can use this structure to effectively characterize and compensate the uncertainties of the system. To this end, we assume that the parameter matrices in (3.25) are subject to some unknown-but-bounded uncertainties as

$$\begin{aligned} \hat{\mathcal{K}}_i &= \mathcal{K}_i + \Delta\mathcal{K}_i, \quad \hat{\mathcal{D}}_i = \mathcal{D}_i + \Delta\mathcal{D}_i \\ \hat{\mathcal{H}}_i &= \mathcal{H}_i + \Delta\mathcal{H}_i, \quad \hat{\boldsymbol{\zeta}}_i^* = \boldsymbol{\zeta}_i^* + \Delta\boldsymbol{\zeta}_i^* \end{aligned} \tag{3.26}$$

where $\hat{\mathcal{K}}_i$, $\hat{\mathcal{D}}_i$, $\hat{\mathcal{H}}_i$, $\hat{\boldsymbol{\zeta}}_i^*$ are the estimations, $\mathcal{K}_i$, $\mathcal{D}_i$, $\mathcal{H}_i$, $\boldsymbol{\zeta}_i^*$ are the nominal values, and $\Delta\mathcal{K}_i$, $\Delta\mathcal{D}_i$, $\Delta\mathcal{H}_i$, $\Delta\boldsymbol{\zeta}_i^*$ correspond to the uncertainties, for $i \in \mathcal{I}_N$. From (3.22), (3.25) and (3.26), the uncertain LPV robot model can be described as

$$\boldsymbol{x}_{k+1} = \hat{A}(\eta)\boldsymbol{x}_k + \hat{B}(\eta)\boldsymbol{u}_k + \hat{\boldsymbol{\zeta}}(\eta) + \boldsymbol{w}_k \tag{3.27}$$

with

$$\begin{bmatrix} \hat{A}(\eta) & \hat{B}(\eta) & \hat{\boldsymbol{\zeta}}(\eta) \end{bmatrix} = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) \begin{bmatrix} \hat{A}_i & \hat{B}_i & \hat{\boldsymbol{\zeta}}_i \end{bmatrix} \tag{3.28}$$

where the local uncertain matrices are defined as

$$\begin{aligned} &\hat{A}_i = A_i + \Delta A_i, && \hat{B}_i = B_i + \Delta B_i, \quad \hat{\boldsymbol{\zeta}}_i = \boldsymbol{\zeta}_i + \Delta\boldsymbol{\zeta}_i \\ &\Delta A_i = \begin{bmatrix} 0 & 0 \\ \Delta\mathcal{K}_i & \Delta\mathcal{D}_i \end{bmatrix}, && \Delta B_i = \begin{bmatrix} 0 \\ \Delta\mathcal{H}_i \end{bmatrix}, \quad \Delta\boldsymbol{\zeta}_i = \begin{bmatrix} 0 \\ \Delta\boldsymbol{\zeta}_i^* \end{bmatrix}. \end{aligned}$$

Exploiting the specific structures of the state-space matrices in (3.25), it follows from (3.27) that

$$\boldsymbol{v}_{r,k+1} = \hat{\mathcal{K}}(\eta)\boldsymbol{q}_{r,k} + \hat{\mathcal{D}}(\eta)\boldsymbol{v}_{r,k} + \hat{\mathcal{H}}(\eta)\boldsymbol{u}_k + \hat{\boldsymbol{\zeta}}^*(\eta) + \boldsymbol{w}_{v,k} \tag{3.29}$$

with

$$\begin{bmatrix} \hat{\mathcal{K}}(\eta) & \hat{\mathcal{D}}(\eta) \\ \hat{\mathcal{H}}(\eta) & \hat{\boldsymbol{\zeta}}^*(\eta) \end{bmatrix} = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) \begin{bmatrix} \hat{\mathcal{K}}_i & \hat{\mathcal{D}}_i \\ \hat{\mathcal{H}}_i & \hat{\boldsymbol{\zeta}}_i^* \end{bmatrix}. \tag{3.30}$$

Note that for (3.29) the disturbance vector $\boldsymbol{w}_k$ is partitioned as $\boldsymbol{w}_k = \begin{bmatrix} \boldsymbol{w}_{q,k}^\top & \boldsymbol{w}_{v,k}^\top \end{bmatrix}^\top$, where the disturbance $\boldsymbol{w}_{q,k} \in \mathbb{R}^r$ (respectively $\boldsymbol{w}_{v,k} \in \mathbb{R}^r$) affects the dynamics of the generalized displacements $\boldsymbol{q}_{r,k}$ (respectively generalized velocities $\boldsymbol{v}_{r,k}$). To represent the parametric uncertainties involved in soft robots modeling, let us define a lumped disturbance

$$\boldsymbol{d}_{e,k} = \mathcal{H}(\eta)^\dagger \Delta\Sigma_k \tag{3.31}$$

where $\Delta\Sigma_k = \mathcal{K}(\eta)\boldsymbol{q}_k + \Delta\mathcal{D}(\eta)\boldsymbol{v}_k + \Delta\mathcal{H}(\eta)\boldsymbol{u}_k + \Delta\boldsymbol{\zeta}^*(\eta)$, and $\mathcal{H}(\eta)^\dagger$ is the pseudo-inverse of $\mathcal{H}(\eta)$. Using the disturbance expression (3.31) and considering (3.29), the uncertain LPV model (3.27) can be reformulated as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A(\eta)\boldsymbol{x}_k + B(\eta)(\boldsymbol{u}_k + \boldsymbol{d}_{e,k}) + \boldsymbol{\zeta}(\eta) + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C\boldsymbol{x}_k. \end{aligned} \tag{3.32}$$

Hence, model (3.32) can be used for LPV control design of soft robots under both disturbances and parametric uncertainties.

Remark 9. Using the projector T_r in (3.10) allows to reduce significantly the model order while preserving the mechanical structure of the original robot model (2.1), i.e., the reduced-order states $\boldsymbol{q}_{r,k}$ and $\boldsymbol{v}_{r,k}$ still respectively play the role of the generalized displacements and velocities with

$$\boldsymbol{q}_{r,k+1} = \boldsymbol{q}_{r,k} + h\boldsymbol{v}_{r,k+1}. \tag{3.33}$$

With such a structure preservation, the robot parameters are only involved in the lower halves of the state-space matrices A_i, B_i and the affine term $\boldsymbol{\zeta}_i$. This leads to two major advantages for LPV modeling of soft robots. First, for RBF network training, the number of parameters to be interpolated is reduced by 50% since the upper halves of the matrices A_{ri}, B_{ri} in (3.14) and the affine term $\boldsymbol{\zeta}_{ri}$ in (3.18) are fixed constants. Second, the modeling uncertainties are only present in the lower

halves of ΔA_i, ΔB_i and $\Delta\boldsymbol{\zeta}_i$, for $i \in \mathcal{I}_N$. This allows to represent the parametric uncertainty and a part of MOR errors in the LPV robot model (3.32) via the lumped disturbance $\boldsymbol{d}_{e,k}$, defined in (3.31). Note that $\boldsymbol{d}_{e,k}$ shares the same distribution matrix $B(\eta)$ with the control input $\boldsymbol{u}_k$, i.e., $\boldsymbol{d}_{e,k}$ is a matching disturbance. This key feature enables an effective EID-based framework for LPV tracking control of soft robots under modeling uncertainty as recently shown in [12]. It is important to note that these advantages of the proposed LPV modeling cannot be achieved with existing POD-based modeling results [48], [79], [80] and related references therein, which cannot directly guarantee the structure preservation.

3.3 Tracking Control Problem Formulation

This section formulates the tracking control problem of soft robots following the same structure and ideas than the linear case presented in the previous chapter. To this end, we define the tracking error as

$$\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}_{r,k} \tag{3.34}$$

with $\boldsymbol{x}_{r,k} = \begin{bmatrix}\boldsymbol{x}_{rq,k}^\top & \boldsymbol{x}_{rv,k}^\top\end{bmatrix}^\top$, where $\boldsymbol{x}_{rq,k} \in \mathbb{R}^r$ (respectively $\boldsymbol{x}_{rv,k} \in \mathbb{R}^r$) is the reference trajectory corresponding to the generalized displacements (respectively velocities). For soft robots, the reference trajectory of the state is not as easy to obtain as the joint angles of rigid-body robots. Since this is the research topic of the next chapter, here we assume that the reference trajectories are directly given. Then, the tracking error dynamics can be defined from (3.32) and (3.34) as

$$\begin{aligned}\boldsymbol{e}_{k+1} = {} & A(\eta)\boldsymbol{e}_k + B(\eta)(\boldsymbol{u}_k + \boldsymbol{d}_{e,k}) + \boldsymbol{w}_k \\ & + A(\eta)\boldsymbol{x}_{r,k} - \boldsymbol{x}_{r,k+1} + \boldsymbol{\zeta}(\eta).\end{aligned} \tag{3.35}$$

To improve the tracking performance of soft robots under modeling uncertainties, we extend the EID-based linear control scheme in [12] to the LPV model (3.35). This feedback-feedforward control scheme is composed of three components as

$$\boldsymbol{u}_k = \boldsymbol{u}_k^{\mathrm{ff}} + \boldsymbol{u}_k^{\mathrm{dc}} + \boldsymbol{u}_k^{\mathrm{fb}} \tag{3.36}$$

where $\boldsymbol{u}_k^{\mathrm{ff}}$ is the feedforward control, $\boldsymbol{u}_k^{\mathrm{dc}}$ is the disturbance-estimator based control, and $\boldsymbol{u}_k^{\mathrm{fb}}$ is the feedback control. The proposed control scheme is illustrated in Figure 3.5.

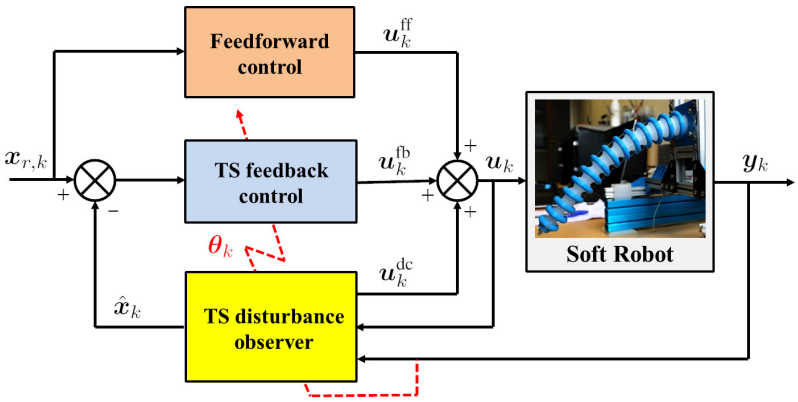


Figure 3.5: EID-based LPV tracking control scheme for elastic soft robots.

3.3.1 Feedforward Control

The approximation capabilities of LPV modeling are substantially improved with the offset terms [81], [82]. However, the presence of these affine terms leads to more complicated stability analysis and control synthesis procedures [83], [84]. In addition, for tracking control, the information of the reference signal can be obtained online. Hence, dealing with the reference trajectory as an unknown input or disturbance may lead to conservative control results. To avoid these issues, we can design a feedforward control $\boldsymbol{u}_k^{\mathrm{ff}}$ to account for the effects of $\boldsymbol{x}_{r,k}$ and $\boldsymbol{\zeta}(\eta)$ on the closed-loop system by exploiting the mechanical structure of the proposed LPV model (3.32) for soft robots, see Remark 9.

The feedforward control is determined under a perfect tracking condition, i.e., $\boldsymbol{e}_k \to 0$ for $k \to \infty$, that is

$$\boldsymbol{x}_k^r = \boldsymbol{x}_{r,k}, \quad \boldsymbol{\theta}_k^r = \boldsymbol{y}_k^r = C\boldsymbol{x}_{r,k}, \quad k \to \infty \tag{3.37}$$

where $\boldsymbol{x}_k^r$, $\boldsymbol{y}_k^r$ and $\boldsymbol{\theta}_k^r$ respectively represent the reduced robot state, the robot output and the scheduling variable corresponding to the perfect tracking. To guarantee a smooth control action for soft robots, especially in large deformation scenarios, the feedforward control is designed following the RBF-based interpolation similar to the construction of the LPV robot model (3.22). From the local viewpoint, the error dynamics (3.35) around the ith equilibrium point is given by

$$\begin{aligned} \boldsymbol{e}_{k+1} = {} & A_i\boldsymbol{e}_k + B_i(\boldsymbol{u}_k + \boldsymbol{d}_{e,k}) + \boldsymbol{w}_k \\ & + A_i\boldsymbol{x}_{r,k} - \boldsymbol{x}_{r,k+1} + \boldsymbol{\zeta}_i. \end{aligned} \tag{3.38}$$

Then, the local feedforward control action can be selected from (3.38) as

$$B_i \boldsymbol{u}_{i,k}^{\mathrm{ff}} + A_i \boldsymbol{x}_{r,k} - \boldsymbol{x}_{r,k+1} + \boldsymbol{\zeta}_i = 0. \tag{3.39}$$

Substituting (3.25) into (3.39), it follows that

$$\mathcal{H}_i \boldsymbol{u}_{i,k}^{\mathrm{ff}} + \mathcal{K}_i \boldsymbol{x}_{rq,k} + \mathcal{D}_i \boldsymbol{x}_{rv,k} - \boldsymbol{x}_{rv,k+1} - \mathcal{K}_i \boldsymbol{q}_{ri}^{*} - \mathcal{H}_i \boldsymbol{u}_i^{*} = 0. \tag{3.40}$$

Under condition (3.37), it follows that

$$\mathcal{K}_i \boldsymbol{x}_{rq,k} \simeq \mathcal{K}_i \boldsymbol{q}_{ri}^{*}. \tag{3.41}$$

Moreover, with a small sampling time value h and a smooth reference trajectory $\boldsymbol{x}_{r,k}$, it follows that

$$\boldsymbol{x}_{r,k+1} \simeq \boldsymbol{x}_{r,k}, \quad \mathcal{D}_i = I - hTS_i^{-1}D_iT^{\top} \simeq I. \tag{3.42}$$

From (3.40), (3.41) and (3.42), it is reasonable to select a local feedforward control as $\boldsymbol{u}_{i,k}^{\mathrm{ff}} = \boldsymbol{u}_i^{*}$. Then, the feedforward control $\boldsymbol{u}_k^{\mathrm{ff}}$ is computed using the RBF-based interpolation as

$$\boldsymbol{u}_k^{\mathrm{ff}} = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) \boldsymbol{u}_{i,k}^{\mathrm{ff}} = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) \boldsymbol{u}_i^{*}. \tag{3.43}$$

Remark 10. A natural solution to obtain the feedforward control $\boldsymbol{u}_k^{\mathrm{ff}}$ is directly based on the tracking error model (3.35) as

$$B(\eta) \boldsymbol{u}_k^{\mathrm{ff}} + A(\eta) \boldsymbol{x}_{r,k} - \boldsymbol{x}_{r,k+1} + \boldsymbol{\zeta}(\eta) = 0. \tag{3.44}$$

Substituting the explicit expressions of $A(\eta)$, $B(\eta)$ and $\boldsymbol{\zeta}(\eta)$ into (3.44), it follows that

$$\mathcal{H}(\eta) \boldsymbol{u}_k^{\mathrm{ff}} + \mathcal{K}(\eta) \boldsymbol{x}_{rq,k} + \mathcal{D}(\eta) \boldsymbol{x}_{rv,k} - \boldsymbol{x}_{rv,k+1} + \boldsymbol{\zeta}^{*}(\eta) = 0. \tag{3.45}$$

Then, the feedforward control can be deduced from (3.45) as

$$\boldsymbol{u}_k^{\mathrm{ff}} = -\mathcal{H}(\eta)^{\dagger} \left(\mathcal{K}(\eta) \boldsymbol{x}_{rq,k} + \mathcal{D}(\eta) \boldsymbol{x}_{rv,k} - \boldsymbol{x}_{rv,k+1} + \boldsymbol{\zeta}^{*}(\eta) \right). \tag{3.46}$$

However, the expression of $\boldsymbol{u}_k^{\mathrm{ff}}$ in (3.46) requires an online pseudo-inverse of the parameter-dependent matrix $\mathcal{H}(\eta)$, i.e., control direction matrix, which could yield large control oscillations or even unstable control behaviors in case of large deformations. On the contrary, the feedforward control $\boldsymbol{u}_k^{\mathrm{ff}}$ in (3.43) is a convex combination of the control input values corresponding to different selected equilibriums, which allows a smooth shifting between operating points of soft robots. Note also that the approximation errors caused by (3.41) and (3.42) when computing $\boldsymbol{u}_k^{\mathrm{ff}}$ in (3.43) can be viewed as a matching disturbance, which can be effectively dealt with using the EID-based control concept as shown in the design of disturbance-estimator based control.

3.3.2 Disturbance-Estimator Based Control

The lumped disturbance $\boldsymbol{d}_{e,k}$, defined in (3.31), enters in the tracking error dynamics (3.35) via the same channel as the control input $\boldsymbol{u}_k$. Then, the disturbance-estimator based control can be designed as follows:

$$\boldsymbol{u}_k^{\mathrm{dc}} = -\hat{\boldsymbol{d}}_k^{\mathrm{ef}} \tag{3.47}$$

where $\hat{\boldsymbol{d}}_k^{\mathrm{ef}}$ is a filtered estimate of the lumped disturbance $\boldsymbol{d}_{e,k}$. Due to the low-frequency behaviors of soft robots, it has been shown that $\boldsymbol{d}_{e,k}$ can be considered of low-frequency [12]. After several trials, the dynamics of $\boldsymbol{d}_{e,k}$ can be sufficiently described with a second-order polynomial signal. Moreover, based on the concept of EID control approach [57], the following low-pass filter is integrated to limit the angular-frequency band of the disturbance estimation:

$$\mathbf{F}(s) = \frac{1}{1+T_f s} I \tag{3.48}$$

where s is the Laplace variable. The filter time constant T_f in (3.48) can be chosen such that

$$T_f \in \left[\frac{1}{10\omega_r}, \frac{1}{5\omega_r}\right],$$

where ω_r is the highest angular frequency selected for disturbance rejection [57]. Then, the continuous-time model of the lumped disturbance is given by [12]

$$\dot{\boldsymbol{d}}(t) = J_c \boldsymbol{d}(t) \tag{3.49}$$

with

$$\boldsymbol{d}(t) = \begin{bmatrix} \boldsymbol{d}^{\mathrm{ef}}(t) \\ \boldsymbol{d}^{\mathrm{e}}(t) \\ \dot{\boldsymbol{d}}^{\mathrm{e}}(t) \end{bmatrix}, \quad J_c = \begin{bmatrix} -\frac{1}{T_f} I & \frac{1}{T_f} I & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix}.$$

Using the explicit Euler-discretization, the discrete-time counterpart of model (3.49) can be obtained as

$$\boldsymbol{d}_{k+1} = J \boldsymbol{d}_k \tag{3.50}$$

with $J = I + hJ_c$. From the LPV robot model (3.32) and the disturbance model (3.50), we propose the following Luenberger-like observer to estimate simultaneously the state $\boldsymbol{x}_k$ and the unknown disturbance $\boldsymbol{d}_{e,k}$:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1} &= A(\eta)\hat{\boldsymbol{x}}_k + B(\eta)(\boldsymbol{u}_k + \hat{\boldsymbol{d}}_{e,k}) + L_x(\eta)(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k) \\ \hat{\boldsymbol{d}}_{k+1} &= J\hat{\boldsymbol{d}}_k + L_d(\eta)(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k) \qquad (3.51) \\ \hat{\boldsymbol{y}}_k &= C\hat{\boldsymbol{x}}_k \qquad (3.52) \end{aligned}$$

where $\hat{\boldsymbol{x}}_k$ is the estimate of $\boldsymbol{x}_k$, and $\hat{\boldsymbol{d}}_k$ is the estimate of $\boldsymbol{d}_k$. The parameter-dependent observer gains $L_x(\eta) \in \mathbb{R}^{n\times q}$ and $L_d(\eta) \in \mathbb{R}^{3p\times q}$ are to be designed. The estimation error dynamics can be defined from (3.32), (3.50) and (3.51) as

$$\boldsymbol{\varepsilon}_{k+1} = (A_o(\eta) - L(\eta)C_o)\boldsymbol{\varepsilon}_k + B_{ow}\boldsymbol{w}_k \tag{3.53}$$

where $\boldsymbol{\varepsilon}_k = \begin{bmatrix}\boldsymbol{\varepsilon}_{x,k}^\top & \boldsymbol{\varepsilon}_{d,k}^\top\end{bmatrix}^\top$, with $\boldsymbol{\varepsilon}_{x,k} = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k$ and $\boldsymbol{\varepsilon}_{d,k} = \boldsymbol{d}_k - \hat{\boldsymbol{d}}_k$. The system matrices in (3.53) are defined as

$$\begin{bmatrix}A_o(\eta) & L(\eta)\end{bmatrix} = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) \begin{bmatrix}A_{oi} & L_i\end{bmatrix},$$

with

$$A_{oi} = \begin{bmatrix}A_i & B_{oi}\\ 0 & J\end{bmatrix}, \qquad B_{ow} = \begin{bmatrix}I\\ 0\end{bmatrix}, \qquad L_i = \begin{bmatrix}L_{xi}\\ L_{di}\end{bmatrix} \tag{3.54}$$

$$C_o = \begin{bmatrix}C & 0\end{bmatrix}, \qquad B_{oi} = \begin{bmatrix}B_i & 0\end{bmatrix}. \tag{3.55}$$

Remark 11. Disturbance observers have been successfully applied to compensate unknown disturbances/uncertainties for tracking control of various engineering systems [85], including robotics applications [86]–[89]. However, related works generally imply complex analysis and design methods or require restrictive assumptions and/or additional measurements, e.g., both system state and its derivatives must be available [85]. Using the generalized Luenberger observer (3.51), we only require the output information to compute directly the disturbance-estimator based control (3.47) with a simple assumption of low-frequency matching disturbance, which is reasonable for soft robots control [12].

3.3.3 Feedback Control

The feedback control is used to guarantee the closed-loop stability under the effects of modeling uncertainties and disturbances. For tracking control purposes, we consider the following LPV proportional-integral control structure:

$$\boldsymbol{u}_k^{\text{fb}} = K_P(\eta)(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_{r,k}) + K_I(\eta)\boldsymbol{e}_{I,k} \tag{3.56}$$

where the parameter-dependent feedback gains $K_P(\eta) \in \mathbb{R}^{p\times n}$ and $K_I(\eta) \in \mathbb{R}^{p\times q}$ are to be determined, and

$$\boldsymbol{e}_{I,k+1} = \boldsymbol{e}_{I,k} + hC(\boldsymbol{x}_k - \boldsymbol{x}_{r,k}). \tag{3.57}$$

With $\boldsymbol{u}_k^{\text{ff}}$, $\boldsymbol{u}_k^{\text{dc}}$ and $\boldsymbol{u}_k^{\text{fb}}$ respectively defined in (3.43), (3.47) and (3.56), substituting the control expression (3.36) into (3.35), the tracking error dynamics can be represented by

$$\boldsymbol{\xi}_{k+1} = (A_c(\eta) + B_c(\eta)K(\eta))\boldsymbol{\xi}_k + B_e(\eta)\boldsymbol{\varepsilon}_k + B_{cw}\boldsymbol{w}_k \tag{3.58}$$

with $\boldsymbol{\xi}_k = \begin{bmatrix}\boldsymbol{e}_k^\top & \boldsymbol{e}_{I,k}^\top\end{bmatrix}^\top$, and

$$\Phi(\eta) = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k)\Phi_i, \quad \Phi \in \{A_c, B_c, K\}$$
$$B_e(\eta) = \sum_{i=1}^{N}\sum_{j=1}^{N} \eta_i(\boldsymbol{\theta}_k)\eta_j(\boldsymbol{\theta}_k)B_{eij}.$$

The system matrices in (3.58) are given by

$$A_{ci} = \begin{bmatrix} A_i & 0 \\ C & I \end{bmatrix}, \qquad B_{ci} = \begin{bmatrix} B_i \\ 0 \end{bmatrix}, \quad B_{cw} = \begin{bmatrix} I \\ 0 \end{bmatrix}$$
$$B_{eij} = \begin{bmatrix} -B_iK_{Pj} & 0 \\ 0 & 0 \end{bmatrix}, \quad K_i = \begin{bmatrix} K_{Pi} & K_{Ii} \end{bmatrix}.$$

Then, the extended closed-loop dynamics can be defined from (3.53) and (3.58) as

$$\bar{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \bar{A}_c(\eta) & B_e(\eta) \\ 0 & \bar{A}_o(\eta) \end{bmatrix} \bar{\boldsymbol{x}}_k + \begin{bmatrix} B_{cw} \\ B_{ow} \end{bmatrix} \boldsymbol{w}_k \tag{3.59}$$

with $\bar{\boldsymbol{x}}_k = \begin{bmatrix}\boldsymbol{\xi}_k^\top & \boldsymbol{\varepsilon}_k^\top\end{bmatrix}^\top$, and

$$\bar{A}_{oi} = A_{oi} - L_iC_o, \qquad \bar{A}_o(\eta) = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k)\bar{A}_{oi}$$
$$\bar{A}_{cij} = A_{ci} + B_{ci}K_j, \quad \bar{A}_c(\eta) = \sum_{i=1}^{N}\sum_{j=1}^{N} \eta_i(\boldsymbol{\theta}_k)\eta_j(\boldsymbol{\theta}_k)\bar{A}_{cij}.$$

To study the stability of system (3.59), we consider the following parameter dependent Lyapunov candidate function:

$$\mathcal{V}(\bar{\boldsymbol{x}}) = \bar{\boldsymbol{x}}^\top \text{diag}\{\lambda Q^{-\top}P_c(\eta)Q^{-1}, P_o(\eta)\}\bar{\boldsymbol{x}} \tag{3.60}$$

where $\lambda > 0$, the matrix $Q \in \mathbb{R}^{(n+q)\times(n+q)}$ is nonsingular, and the parameter-dependent matrices $P_c(\eta) \in \mathbb{R}^{(n+q)\times(n+q)}$ and $P_o(\eta) \in \mathbb{R}^{(n+3p)\times(n+3p)}$ are defined as

$$P_c(\eta) = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) P_{ci}, \quad P_o(\eta) = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_k) P_{oi},$$

with $P_{ci} \succ 0$ and $P_{oi} \succ 0$, for $\forall i \in \mathcal{I}_N$. For tracking control purposes, we also introduce the performance output $\boldsymbol{z}_k$ associated to system (3.59) as the position tracking error $\boldsymbol{z}_k = \boldsymbol{e}_k = C_z \boldsymbol{\xi}_k$, with $C_z = \begin{bmatrix} C & 0 \end{bmatrix}$, or

$$\boldsymbol{z}_k = F \bar{\boldsymbol{x}}_k, \quad F = \begin{bmatrix} C_z & 0 \end{bmatrix}. \tag{3.61}$$

We are ready to formulate the following observer-based output tracking control problem for soft robots.

Problem 2. Consider the LPV robot model (3.32) with the control law (3.36). Determine the parameter-dependent observer gain $L(\boldsymbol{\theta})$ and controller gain $K(\boldsymbol{\theta})$ such that the extended error dynamics (3.59) verifies following properties.

(P1) If $\boldsymbol{w}_k = 0$, for $\forall k \in \mathbb{N}$, the closed-loop dynamics (3.59) is exponentially stable with a decay rate $\alpha \in (0, 1)$.

(P2) The closed-loop state $\bar{\boldsymbol{x}}_k$ is uniformly bounded for any initial condition $\bar{\boldsymbol{x}}_0$ and any sequence $\{\boldsymbol{w}_k\}_{k\in\mathbb{N}} \in \ell_\infty$. That is, there exists a bound $\varphi(\bar{\boldsymbol{x}}_0, \|\boldsymbol{w}\|_{\ell_\infty})$ such that $\|\bar{\boldsymbol{x}}_k\| \leq \varphi(\bar{\boldsymbol{x}}_0, \|\boldsymbol{w}\|_{\ell_\infty})$, for $\forall k \geq 0$. Moreover, the performance output verifies

$$\lim_{k\to\infty} \sup \|\boldsymbol{z}_k\| < \gamma \|\boldsymbol{w}\|_{\ell_\infty} \tag{3.62}$$

where the $\ell_\infty-$gain γ is specified in Theorem 2. We also deduce from (3.62) that if $\bar{\boldsymbol{x}}_0 = 0$, then $\|\boldsymbol{z}_k\| < \gamma \|\boldsymbol{w}\|_{\ell_\infty}$, for $\forall k \in \mathbb{N}$.

System (3.59) verifying properties (P1)–(P2) is globally uniformly $\ell_\infty-$stable with a performance level γ, see [90, Chapter 4]. It follows from (3.61) and (3.62) that a smaller value of the $\ell_\infty-$gain γ leads to a better tracking control performance. Note also that a larger value of the decay rate α leads to a faster closed-loop response of the soft robot, which may induce aggressive control behaviors.

3.4 LPV Output Feedback Tracking Control with $\ell_\infty-$Gain Performance Guarantee

This section presents LMI-based conditions to simultaneously design an LPV observer (3.51) and an LPV feedback controller (3.56) such that system (3.59) verifies the closed-loop properties specified in Problem 2.

Theorem 2. Consider the LPV robot model (3.32) with the control law (3.36) and a decay rate $\alpha \in (0,1)$. If there exist parameter-dependent positive definite matrices $P_o(\eta) \in \mathbb{R}^{(n+3p)\times(n+3p)}$, $P_c(\eta) \in \mathbb{R}^{(n+q)\times(n+q)}$, parameter-dependent matrices $M(\eta) \in \mathbb{R}^{p\times(n+q)}$, $N(\eta) \in \mathbb{R}^{(n+3p)\times q}$, matrices $G \in \mathbb{R}^{(n+3p)\times(n+3p)}$, $Q \in \mathbb{R}^{(n+q)\times(n+q)}$, and positive scalars μ, ν such that

$$\begin{bmatrix} (1-\alpha)P_o(\eta) & 0 & A_o^\top(\eta)G^\top - C_o^\top N^\top(\eta) \\ \star & \alpha\nu I & B_{ow}^\top G^\top \\ \star & \star & G + G^\top - P_o(\eta_+) \end{bmatrix} \succ 0 \tag{3.63}$$

$$\begin{bmatrix} (1-\alpha)P_c(\eta) & A_c(\eta)Q + B_c(\eta)M(\eta) \\ \star & Q + Q^\top - P_c(\eta_+) \end{bmatrix} \succ 0 \tag{3.64}$$

$$\begin{bmatrix} P_c(\eta) & 0 & Q^\top C_z^\top \\ \star & P_o(\eta) & 0 \\ \star & \star & \mu I \end{bmatrix} \succeq 0 \tag{3.65}$$

for all $\eta(\boldsymbol{\theta}_k), \eta(\boldsymbol{\theta}_{k+1}) \in \Omega$, with

$$P_o(\eta_+) = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_{k+1}) P_{oi}, \quad P_c(\eta_+) = \sum_{i=1}^{N} \eta_i(\boldsymbol{\theta}_{k+1}) P_{ci}.$$

Then, the closed-loop system (3.59) verifies the properties defined in Problem 2 with a guaranteed $\ell_\infty-$gain $\gamma = \sqrt{\nu\mu}$. Moreover, the feedback control gain and the observer gain are respectively given by

$$K(\eta) = M(\eta)Q^{-1}, \quad L(\eta) = G^{-1}N(\eta). \tag{3.66}$$

Proof. To begin with, it follows from (3.63) and (3.64) that

$$G + G^\top \succ P_o(\eta_+), \quad Q + Q^\top \succ P_c(\eta_+). \tag{3.67}$$

Since $P_o(\eta_+) \succ 0$ and $P_c(\eta_+) \succ 0$, for $\forall \eta(\boldsymbol{\theta}_{k+1}) \in \Omega$, it follows from (3.67) that the matrices G and Q are nonsingular, i.e., invertible, thus the feedback control gain

$K(\eta)$ and the observer gain matrix $L(\eta)$ in (3.66) are well-defined. For brevity, we denote

$$\begin{aligned}
\bar{\mathbf{A}}_o(\eta) &= \begin{bmatrix}\bar{A}_o(\eta) & B_{ow}\end{bmatrix} \\
\mathbf{P}_o(\eta) &= \text{diag}\{(1-\alpha)P_o(\eta), \alpha\nu I\} \\
\mathbf{A}_c(\eta) &= A_c(\eta)Q + B_c(\eta)M(\eta) \\
\mathbf{Q}_c(\eta) &= Q^{-\top}P_c(\eta)Q^{-1}, \quad \mathbf{Q}_c(\eta_+) = Q^{-\top}P_c(\eta_+)Q^{-1}.
\end{aligned}$$

Inspired by the congruence transformation proposed in [91], we multiply inequality (3.63) with

$$\begin{bmatrix} I & 0 & -\bar{A}_o^\top(\eta) \\ 0 & I & -B_{ow}^\top \end{bmatrix}$$

on the left and its transpose on the right while considering $N(\eta) = GL(\eta)$, it follows that

$$\Gamma(\eta, \eta_+) = \bar{\mathbf{A}}_o^\top(\eta)P_o(\eta_+)\bar{\mathbf{A}}_o(\eta) - \mathbf{P}_o(\eta) \prec 0. \tag{3.68}$$

Then, multiplying inequality (3.64) with

$$\begin{bmatrix} I & -\mathbf{A}_c^\top(\eta)Q^{-\top} \end{bmatrix}$$

on the left and its transpose on the right, we have

$$\mathbf{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)\mathbf{A}_c(\eta) - (1-\alpha)P_c(\eta) \prec 0. \tag{3.69}$$

Pre- and post-multiplying (3.69) with $Q^{-\top}$ and its transpose while considering $M(\eta) = K(\eta)Q$, it follows that

$$\Pi(\eta, \eta_+) = \bar{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)\bar{A}_c(\eta) - (1-\alpha)\mathbf{Q}_c(\eta) \prec 0. \tag{3.70}$$

Since $\Gamma(\eta, \eta_+) \prec 0$ and $\Pi(\eta, \eta_+) \prec 0$, then for $\Psi(\eta, \eta_+) \succeq 0$, there always exists a positive scalar λ such that [68]

$$\Gamma(\eta, \eta_+) + \lambda\Psi(\eta, \eta_+) \prec \lambda\Upsilon^\top(\eta, \eta_+)\Pi^{-1}(\eta, \eta_+)\Upsilon(\eta, \eta_+) \tag{3.71}$$

with

$$\begin{aligned}
\Psi(\eta, \eta_+) &= \begin{bmatrix} B_e^\top(\eta) \\ B_c^\top(\eta) \end{bmatrix} \mathbf{Q}_c(\eta_+) \begin{bmatrix} B_e(\eta) & B_c(\eta) \end{bmatrix} \\
\Upsilon(\eta, \eta_+) &= \begin{bmatrix} \bar{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)B_e(\eta) & \bar{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)B_{cw} \end{bmatrix}.
\end{aligned} \tag{3.72}$$

Applying the Schur complement lemma [69], we can prove that condition (3.71) is equivalent to

$$\begin{bmatrix} \lambda\Pi(\eta,\eta_+) & \lambda\Upsilon(\eta,\eta_+) \\ \star & \Gamma(\eta,\eta_+)+\lambda\Psi(\eta,\eta_+) \end{bmatrix} \prec 0. \tag{3.73}$$

Substituting the expressions of $\Gamma(\eta,\eta_+)$ in (3.68), $\Pi(\eta,\eta_+)$ in (3.70), $\Psi(\eta,\eta_+)$ and $\Upsilon(\eta,\eta_+)$ in (3.72) into condition (3.73), we obtain

$$\begin{bmatrix} \Sigma_{11}(\eta,\eta_+) & \Sigma_{12}(\eta,\eta_+) & \Sigma_{13}(\eta,\eta_+) \\ \star & \Sigma_{22}(\eta,\eta_+) & \Sigma_{23}(\eta,\eta_+) \\ \star & \star & \Sigma_{33}(\eta,\eta_+) \end{bmatrix} \prec 0 \tag{3.74}$$

with

$$\begin{aligned}
\Sigma_{11}(\eta,\eta_+) &= \lambda\bar{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)\bar{A}_c(\eta)+\lambda(\alpha-1)\mathbf{Q}_c(\eta)\\
\Sigma_{12}(\eta,\eta_+) &= \lambda\bar{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)B_e(\eta)\\
\Sigma_{13}(\eta,\eta_+) &= \lambda\bar{A}_c^\top(\eta)\mathbf{Q}_c(\eta_+)B_{cw}\\
\Sigma_{22}(\eta,\eta_+) &= \lambda B_e^\top(\eta)\mathbf{Q}_c(\eta_+)B_e(\eta)+\Xi(\eta,\eta_+)\\
\Xi(\eta,\eta_+) &= \bar{A}_o^\top(\eta)P_o(\eta_+)\bar{A}_o(\eta)-(1-\alpha)P_o(\eta)\\
\Sigma_{23}(\eta,\eta_+) &= \lambda B_e(\eta)^\top\mathbf{Q}_c(\eta_+)B_{cw}+\bar{A}_o^\top(\eta)P_o(\eta_+)B_{ow}\\
\Sigma_{33}(\eta,\eta_+) &= -\alpha\nu I+\lambda B_{cw}^\top\mathbf{Q}_c(\eta_+)B_{cw}+B_{ow}^\top P_o(\eta_+)B_{ow}.
\end{aligned}$$

Since $\lambda B_e^\top(\eta)\mathbf{Q}_c(\eta_+)B_e(\eta)\succeq 0$, it follows from (3.74) that

$$\begin{bmatrix} \Sigma_{11}(\eta,\eta_+) & \Sigma_{12}(\eta,\eta_+) & \Sigma_{13}(\eta,\eta_+) \\ \star & \Xi(\eta,\eta_+) & \Sigma_{23}(\eta,\eta_+) \\ \star & \star & \Sigma_{33}(\eta,\eta_+) \end{bmatrix} \prec 0. \tag{3.75}$$

Multiplying inequality (3.75) with $\begin{bmatrix}\bar{\boldsymbol{x}}_k^\top & \boldsymbol{w}_k^\top\end{bmatrix}^\top$ on the left and its transpose on the right, the following condition can be obtained after some algebraic manipulations:

$$\Delta\mathcal{V}(\bar{\boldsymbol{x}}_k)+\alpha\left(\mathcal{V}(\bar{\boldsymbol{x}}_k)-\nu\boldsymbol{w}_k^\top\boldsymbol{w}_k\right)<0,\quad \forall k\in\mathbb{N} \tag{3.76}$$

where the parameter-dependent Lyapunov function $\mathcal{V}(\bar{\boldsymbol{x}}_k)$ is defined in (3.60), and $\Delta\mathcal{V}(\bar{\boldsymbol{x}}_k)=\mathcal{V}(\bar{\boldsymbol{x}}_{k+1})-\mathcal{V}(\bar{\boldsymbol{x}}_k)$ is its difference along the trajectories of the extended closed-loop error dynamics (3.59). We distinguish the two following cases.

First, if $\boldsymbol{w}_k=0$, for $\forall k\in\mathbb{N}$, it follows from (3.76) that

$$\Delta\mathcal{V}(\bar{\boldsymbol{x}}_k)+\alpha\mathcal{V}(\bar{\boldsymbol{x}}_k)<0,\quad \forall k\in\mathbb{N} \tag{3.77}$$

which proves Property (P1) on the exponential stability with a decay rate α of system (3.59).

Second, if $\boldsymbol{w}_k \neq 0$ and $\{\boldsymbol{w}_k\}_{k\in\mathbb{N}} \in \ell_\infty$, it follows from (3.76) that

$$\mathcal{V}(\bar{\boldsymbol{x}}_k) < (1-\alpha)\mathcal{V}(\bar{\boldsymbol{x}}_{k-1}) + \alpha\nu \left\|\boldsymbol{w}_{k-1}\right\|^2, \quad \forall k \geq 1. \tag{3.78}$$

By recursivity and since $\alpha \in (0,1)$, it follows from (3.78) that

$$\begin{aligned}\mathcal{V}(\bar{\boldsymbol{x}}_k) &< (1-\alpha)^k \mathcal{V}(\bar{\boldsymbol{x}}_0) + \alpha\nu \sum_{i=0}^{k-1} (1-\alpha)^i \left\|\boldsymbol{w}_{k-1-i}\right\|^2 \\ &< (1-\alpha)^k \mathcal{V}(\bar{\boldsymbol{x}}_0) + \alpha\nu \|\boldsymbol{w}\|_{\ell_\infty}^2 \sum_{i=0}^{k-1} (1-\alpha)^i \\ &< (1-\alpha)^k \mathcal{V}(\bar{\boldsymbol{x}}_0) + \nu \|\boldsymbol{w}\|_{\ell_\infty}^2, \quad \forall k \geq 1 \end{aligned} \tag{3.79}$$

which guarantees that $\bar{\boldsymbol{x}}_k$ is uniformly bounded for any initial condition $\bar{\boldsymbol{x}}_0$ and any sequence $\{\boldsymbol{w}_k\}_{k\in\mathbb{N}} \in \ell_\infty$.

Applying the congruence transformation $\text{diag}\{Q^{-\top}, I, I\}$ to inequality (3.65), it follows that

$$\begin{bmatrix} \mathbf{Q}_c(\eta) & 0 & C_z^\top \\ \star & P_o(\eta) & 0 \\ \star & \star & \mu I \end{bmatrix} \succeq 0. \tag{3.80}$$

Using the Schur complement lemma, we can prove that condition (3.80) is equivalent to

$$\mu \text{diag}\{\mathbf{Q}_c(\eta), P_o(\eta)\} - F^\top F \succeq 0. \tag{3.81}$$

Pre- and post-multiplying condition (3.81) with $\bar{\boldsymbol{x}}_k^\top$ and its transpose yields

$$\|\boldsymbol{z}_k\|^2 \leq \mu \mathcal{V}(\bar{\boldsymbol{x}}_k). \tag{3.82}$$

It follows from (3.79) and (3.82) that

$$\|\boldsymbol{z}_k\| \leq \sqrt{\mu \mathcal{V}(\bar{\boldsymbol{x}}_0)}(1-\alpha)^{k/2} + \gamma \|\boldsymbol{w}\|_{\ell_\infty}, \quad \forall k \geq 1 \tag{3.83}$$

with $\gamma = \sqrt{\nu\mu}$. For any initial condition $\bar{\boldsymbol{x}}_0$, it follows from (3.83) that

$$\lim_{k\to\infty} \sup \|\boldsymbol{z}_k\| \leq \gamma \|\boldsymbol{w}\|_{\ell_\infty}. \tag{3.84}$$

Conditions (3.79), (3.83) and (3.84) guarantee Property (P2), which concludes the proof. □

Remark 12. Using some congruence matrix transformations, the LPV observer-based control design in Theorem 2 offers an extra degree of freedom to enable less conservative results. Specifically, slack variables Q and G are introduced such that the parameter-dependent matrices $P_c(\eta)$ and $P_o(\eta)$ of the Lyapunov function $\mathcal{V}(\bar{\boldsymbol{x}})$ defined in (3.60) can be decoupled from any product with the state-space matrices. Moreover, the feedback control gain $K(\eta)$ and the observer gain $L(\eta)$, given in (3.66), do not explicitly depend on $P_c(\eta)$ and $P_o(\eta)$, respectively. It is important to note that the design conservatism of the control results in Theorem 2 can be further reduced using parameter-dependent slack variables as $Q(\eta)$ and $G(\eta)$. However, in this case the expressions of the control and observer gains require online matrix inversions, i.e., $K(\eta) = M(\eta)Q^{-1}(\eta)$ and $L(\eta) = G^{-1}(\eta)N(\eta)$, which could induce some difficulties in control tuning for practical uses.

Theorem 2 cannot be directly used for control design due to the presence of $\eta(\boldsymbol{\theta}_k), \eta(\boldsymbol{\theta}_{k+1}) \in \Omega$ in the LMI-based matrix inequalities (3.63)–(3.65). It is important to note that the Gaussian membership functions $\eta_i(\boldsymbol{\theta}_k)$, for $i \in \mathcal{I}_N$, are locally defined. Hence, there is no overlap between two membership functions $\eta_i(\boldsymbol{\theta}_k)$ and $\eta_j(\boldsymbol{\theta}_k)$, which are not adjacent to each other as illustrated in Figure 3.2. As a result, the product of two non adjacent membership functions is identically null, i.e., $\eta_i(\boldsymbol{\theta}_k)\eta_j(\boldsymbol{\theta}_k) = 0$, for $\forall k \in \mathbb{N}$. Exploiting this fact and the LMI relaxation result proposed in [92], the following theorem provides a tractable solution for Problem 2 while reducing the design conservatism and numerical complexity.

Theorem 3. Consider the LPV robot model (3.32) with the control law (3.36) and a decay rate $\alpha \in (0, 1)$. Assume that the maximum number of membership functions that are activated for all $k \in \mathbb{N}$ is less than or equal to s where $1 < s \leq N$. Then, the closed-loop system (3.59) verifies the properties defined in Problem 2 with a guaranteed minimum $\ell_\infty-$gain if there exist positive definite matrices $P_{oi} \in \mathbb{R}^{(n+3p)\times(n+3p)}$, $P_{ci} \in \mathbb{R}^{(n+q)\times(n+q)}$, matrices $M_i \in \mathbb{R}^{p\times(n+q)}$, $N_i \in \mathbb{R}^{(n+3p)\times q}$, for $i \in \mathcal{I}_N$, matrices $G \in \mathbb{R}^{(n+3p)\times(n+3p)}$, $Q \in \mathbb{R}^{(n+q)\times(n+q)}$, a positive semidefinite matrix $W \in \mathbb{R}^{(n+q)\times(n+q)}$, and positive scalars μ, ν, solution to the following optimization

problem:

$$\text{minimize} \quad \mu + \nu, \tag{3.85}$$

$$\text{such that} \tag{3.86}$$

$$\begin{bmatrix} (1-\alpha)P_{oi} & 0 & A_{oi}^\top G^\top - C_o^\top N_i^\top \\ \star & \alpha\nu I & B_{ow}^\top G^\top \\ \star & \star & G + G^\top - P_{ol} \end{bmatrix} \succ 0, \;\; i,l \in \mathcal{I}_N \tag{3.87}$$

$$\begin{bmatrix} P_{ci} & 0 & Q^\top C_z^\top \\ \star & P_{oi} & 0 \\ \star & \star & \mu I \end{bmatrix} \succeq 0, \quad i \in \mathcal{I}_N \tag{3.88}$$

$$\Theta_{iil} \succ (s-1)\mathbf{W}, \quad i \in \mathcal{I}_N \tag{3.89}$$

$$\Theta_{ijl} + \Theta_{jil} \succeq -2\mathbf{W}, \quad i,j,l \in \mathcal{I}_N, \; i < j \tag{3.90}$$

for all i and j excepting the pairs (i,j) such that $\eta_i(\boldsymbol{\theta}_k)\eta_j(\boldsymbol{\theta}_k) = 0$, $\forall k \in \mathbb{N}$ and $s > 1$. The quantity Θ_{ijl} and the matrix $\mathbf{W}$ in (3.89)–(3.90) are defined as

$$\Theta_{ijl} = \begin{bmatrix} (1-\alpha)P_{ci} & A_{ci}Q + B_{ci}M_j \\ \star & Q + Q^\top - P_{cl} \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} W & 0 \\ \star & 0 \end{bmatrix}.$$

Proof. Since $\eta(\boldsymbol{\theta}_k), \eta(\boldsymbol{\theta}_{k+1}) \in \Omega$, for $\forall k \in \mathbb{N}$, multiplying condition (3.87) with $\eta_i(\boldsymbol{\theta}_k)\eta_l(\boldsymbol{\theta}_{k+1}) \geq 0$ and summing up for $i,l \in \mathcal{I}_N$, we obtain (3.63). Similarly, we can prove that condition (3.88) implies (3.65). Moreover, using the relaxation result in [92], it follows that conditions (3.89)–(3.90) guarantee (3.64). Note also that the $\ell_\infty-$gain $\gamma = \sqrt{\mu\nu}$ can be minimized via the optimization problem (3.85). Then, by the result of Theorem 2, we can conclude the proof. □

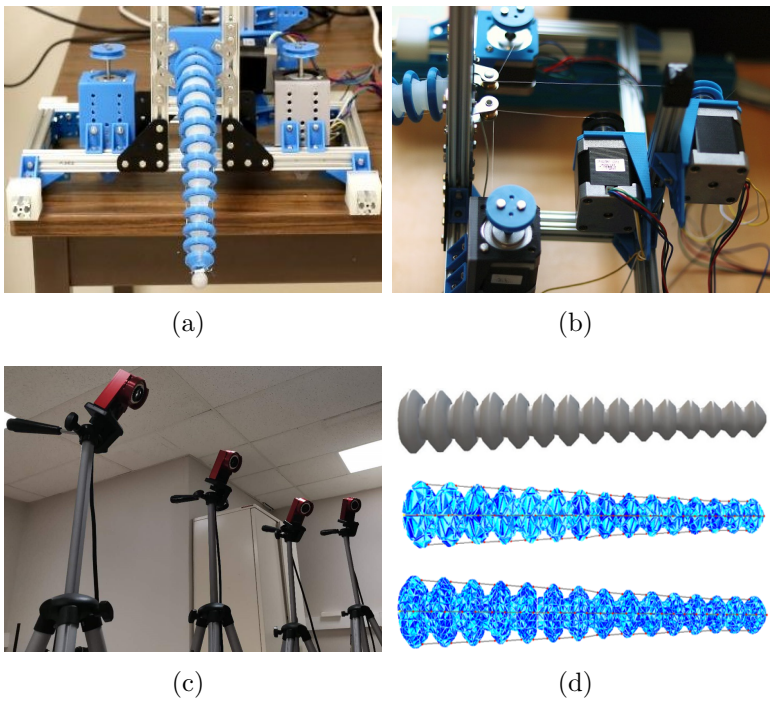
Remark 13. The LPV tracking control design of soft robots is reformulated as a convex optimization problem (3.85) under LMI constraints (3.87)–(3.90), which can be effectively solved with YALMIP toolbox and SDPT3 solver [64].

3.5 Experimental Results of LPV Tracking Control for a Soft Trunk Robot

This section presents the experimental results obtained with a Trunk robot to illustrate the effectiveness of the proposed RBF-based LPV modeling and the EID-based LPV output feedback control method for elastic soft robots.

3.5.1 Experimental Soft Robot Platform

The Trunk robot platform is depicted in Figure 3.6. This soft robot is made of silicone rubber with 14 segments to make it highly deformable. The weight of the Trunk robot is 40 [g] and its length is 195 [mm] in the initial position. This soft robot is driven by four stepper motors via cables mounted on the robot body to guarantee the accessibility of each direction in the workspace. The two control inputs of the robot are realized by pulling these four cables as a pulley system. The position of the end-effector, e.g., system output, is measured by an OptiTrack tracking system and a reflective marker is mounted on the endpoint as an end-effector. The data rate of the OptiTrack tracking system is 100 [Hz] with an accuracy of 0.1 [mm] after preliminary calibrations. Note that for this soft robot, since the center of mass changes with respect to its motion and deformation, the gravity effect plays a role of a time-varying disturbance.



(a) (b)

(c) (d)

Figure 3.6: Soft Trunk robot. (a) Robot platform. (b) Stepper motors. (c) OptiTrack tracking system. (d) FEM modeling of the trunk with different meshes.

3.5.2 LPV Model Validation

The FEM simulation and modeling of the soft Trunk robot is implemented using the open-source SOFA framework[1]. The Trunk robot FEM model has 1484 nodes with 8904 state variables. A four-order reduced model can be obtained via the POD-based order reduction method as shown in the previous chapter. To construct the RBF-based LPV model of the soft Trunk robot, we collect the data of 45 different robot configurations, i.e., equilibrium points, that can cover the whole workspace, see Figure 3.3. Then, POD-based model reduction method is applied to obtain the 45 corresponding reduced-order linear submodels of the form (3.11). After some preliminary validations, 9 of these submodels are selected to build the interpolated LPV model of the Trunk robot using Algorithm 1 as illustrated in Figure 3.3.

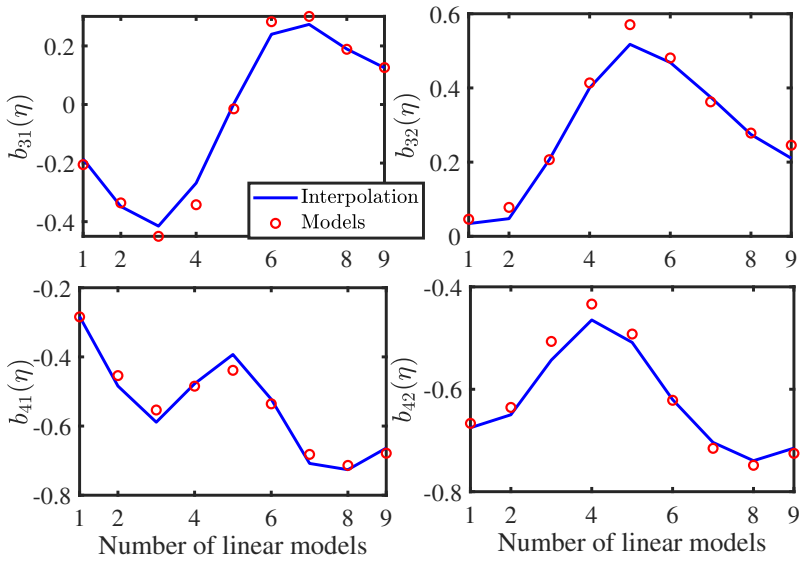


Figure 3.7: Evolution of the non-zero elements of the input matrix $B(\eta)$: RBF-based interpolation results (-), model parameters from collected data (○).

To illustrate the nonlinear phenomenon caused by the actuation forces, Figure

[1]More details on the plugin SoftRobots for the SOFA framework can be found at the address: https://project.inria.fr/softrobot.

3.7 shows the evolution of the non-zero elements of the reduced-order input matrix

$$B(\eta) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_{31}(\eta) & b_{32}(\eta) \\ b_{41}(\eta) & b_{41}(\eta) \end{bmatrix}$$

corresponding to a trajectory with large deformation as illustrated in Figure 3.2. While for the rest of system matrices $A(\eta)$ and C, the parameters either have only a small variation range or are constants.

First, we can see that the parameter values of the 9 local linearized models, used to establish the reduced-order LPV robot model, are well interpolated with the proposed RBF-based method. Second, the value of $b_{31}(\eta)$, corresponding to the action from the horizontal cable actuator to the x-axis of robot, varies from negative to positive. This illustrates the nonlinear behavior of elastic soft robots concerning the change of the actuation direction as previously discussed.

To further validate the proposed LPV modeling method, we compare the following robot models:

- the linear model in [12], corresponding to the equilibrium point $(\boldsymbol{q}_0, \boldsymbol{v}_0, \boldsymbol{u}_0) \equiv (0, 0, 0)$;
- the 1484-node nonlinear FEM model used to derive the proposed LPV robot model;
- the proposed LPV model.

For comparison purposes, we apply a ramp input signal to the left cable of Trunk robot to gradually increase the robot deformation, which can reproduce the nonlinear behavior illustrated in Figure 3.2. The displacement of the robot end-effector is measured. Besides, simulations are performed with the same input signal for the linear model, the FEM model and the LPV model. The comparison result is shown in Figure 3.8.

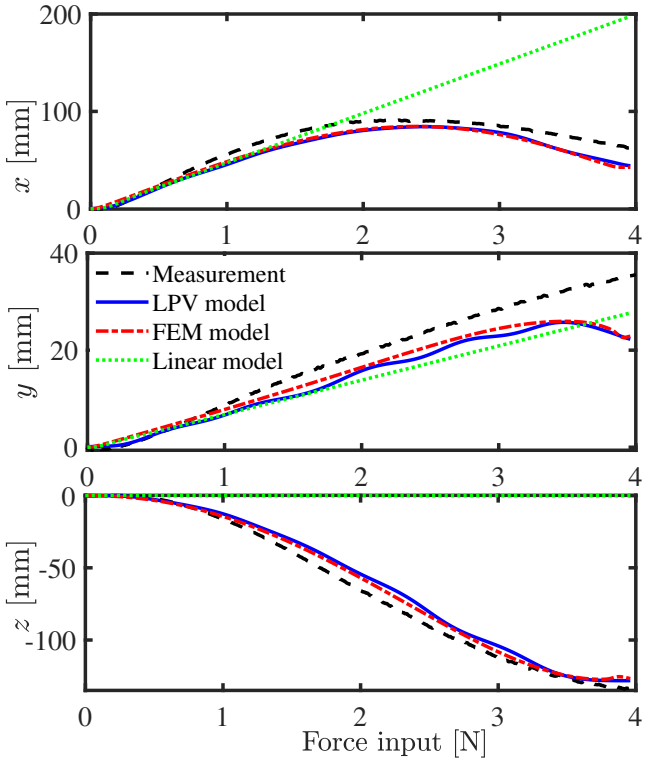


Figure 3.8: Comparison between different modeling methods for the Trunk robot.

Observe that the nonlinear FEM model and the LPV model provide similar behaviors, which capture well the nonlinear dynamics of the soft Trunk robot even if the input signal becomes large. As expected, the linear reduced-order model can only approximate the robot dynamics under small deformations with a small input signal. Note that the linear model is obtained from the configuration, for which the actuation does not have any impact on the $z-$axis displacement. The above validation results in Figures 3.7 and 3.8 confirm the validity of the proposed LPV modeling method. Hereafter, the relevance of this LPV modeling for dynamic tracking control design of soft robots under large deformations is demonstrated.

Although our LPV model captures the nonlinear behavior of the system well, the LPV model is not a strictly equivalent nonlinear model. Since the scheduling variables of the LPV model need to rely on the measurement of the system, the LPV model will synchronize with the real system through the scheduling variables and the Luenberger state observer. In other words, if open-loop simulations are performed directly on the LPV model, small errors in scheduling variables could accumulate and lead to significant model errors at high speeds or oscillations.

3.5.3 Tracking Control Validation

To show the effectiveness of the proposed LPV tracking control framework for elastic soft robots, we perform several experimental tests with both large and small deformations. Note that this soft robot is horizontally positioned as an elephant trunk, which leads to a distorted sphere workspace due to the gravity effect as depicted in Figure 3.9. Since the center of mass changes with respect to the robot motion and deformation, the gravity effect plays a role of time-varying disturbance for the robot control system. For comparison purposes, we have shown in the previous chapter that the linear EID-based controller therein can outperform, in terms of tracking performance, some standard controllers for soft elastic robots, e.g., inverse kinematics based QP control [15], [39] and Jacobian-based PID control [66], [67]. Hence, in the sequel we mainly focus on the comparisons between the proposed LPV control method, the linear EID-based control method in [12] and a linear iterative learning control (ILC) method. This latter is known as a powerful scheme for dynamic tracking control in the presence of non-random disturbances, such as unmodeled nonlinearities of the system [93].

Test 1: Dynamic Tracking with a Predefined Trajectory

For this test, we select a reference trajectory that covers the entire workspace of the Trunk robot. Note that the workspace of the Trunk robot can be parameterized by a spherical coordinate system with two angular coordinates $(\vartheta(t), \varphi(t))$, which enable to easily define the Cartesian coordinates of end effector trajectories in the workspace. For illustrations, we consider a reference trajectory with the following latitude and azimuth angles:

$$\vartheta(t) = 2.1\sin(0.3t)\ [\text{rad}], \quad \varphi(t) = 0.5\cos(0.3t)\ [\text{rad}] \tag{3.91}$$

which corresponds to a large circular trajectory in the workspace. Figure 3.9 depicts the evolution of the end-effector positions of the Trunk robot within the shell-like workspace, which are obtained with the proposed LPV control method and the linear control method in [12]. We can see that the linear control quickly fails when the trajectory goes beyond its operating region of the workspace. In contrast, LPV control can provide an effective tracking for the whole predefined trajectory. To examine the tracking control performance in more detail, Figures 3.10 and 3.11 present the tracking control results, projected on the $\vartheta\varphi-$plane. Remark that both LPV and linear control methods share a similar tracking performance around the initial robot configuration. However, after reaching the singular configuration of the Trunk robot, i.e., with a change of the actuation direction, the linear control method is unable to

perform the tracking task while the LPV control method still offers a satisfactory reference tracking result. The corresponding control inputs are shown in Figure 3.12. Note that the $\vartheta-$axis force control input has a sinus shape, which is synchronized with the $\vartheta-$axis trajectory. However, the $\varphi-$axis control input is similar to a square wave, which does not correspond to the form of the $\varphi-$axis trajectory. This is not the case of the dynamic tracking control with small deformations in [12], which also illustrates the effects of large nonlinearities, i.e., large deformations, in soft robots control. Video of this experiment can be found at: https://bit.ly/3RIXlsF.

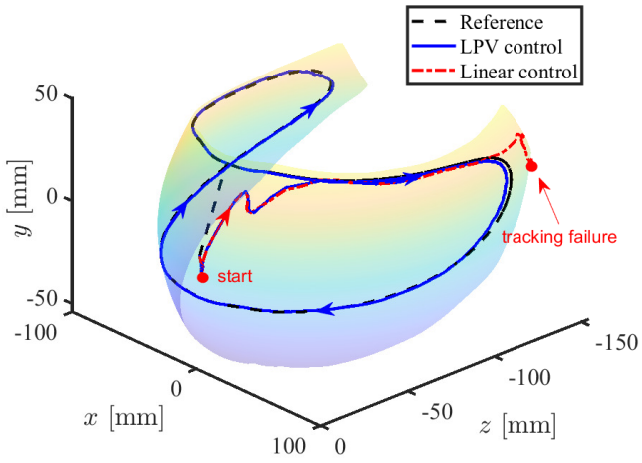


Figure 3.9: Experimental results of 3D dynamic tracking control with a circular reference trajectory (Test 1).

Test 2: Robustness with respect to External Disturbances

The EID-based control concept has been shown in [12] as an effective tool to deal with the parametric uncertainties of elastic soft robots, analyzed in Section 3.2.3, for linear dynamic tracking control. This test is used to demonstrate that this control concept is also useful for interpolated LPV control framework to deal with not only parametric uncertainties but also external disturbances. To this end, we add an extra load near the end-effector of the Trunk robot as an external disturbance during the trajectory tracking as shown in Figure 3.13. The load contains 3 coins with a total mass of 24.8 [g], which is about 15% of the robot weight. The tracking task of this test is composed of two phases with the same reference trajectory as defined in (3.91), and the load is added in the second phase for comparison purposes. The corresponding

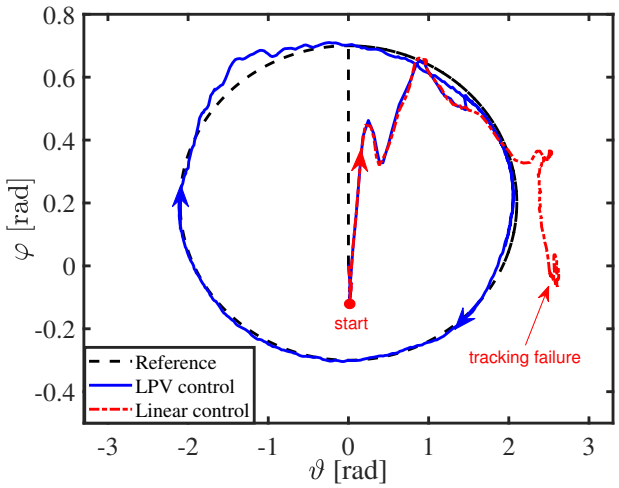


Figure 3.10: Experiment tracking control results with a circular reference trajectory projected on the spherical coordinate $\vartheta\varphi-$plane (Test 1).

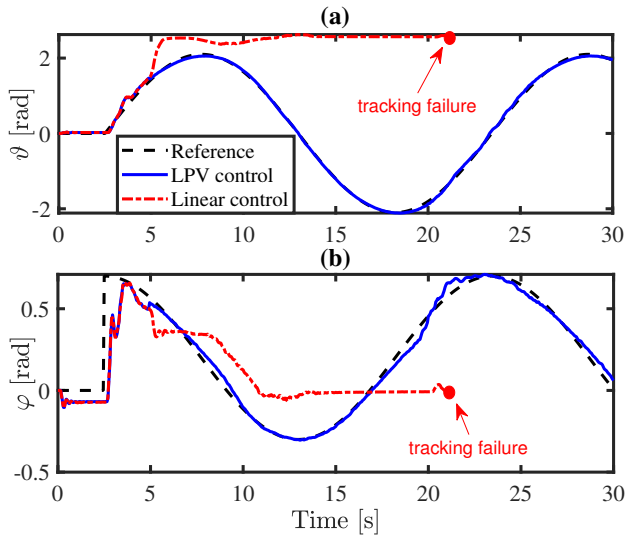


Figure 3.11: Tracking control task in the parametric space (Test 1). (a) Tracking performance along $\vartheta-$axis. (b) Tracking performance along $\varphi-$axis.

tracking control result in 3D and its projection on each axis are presented in Figures 3.14 and 3.15, respectively. We can see that the disturbance effect, caused by the extra load added at around 27 [s], is quickly compensated. Indeed, there is no significant difference after about 1 [s] on the tracking control performance between

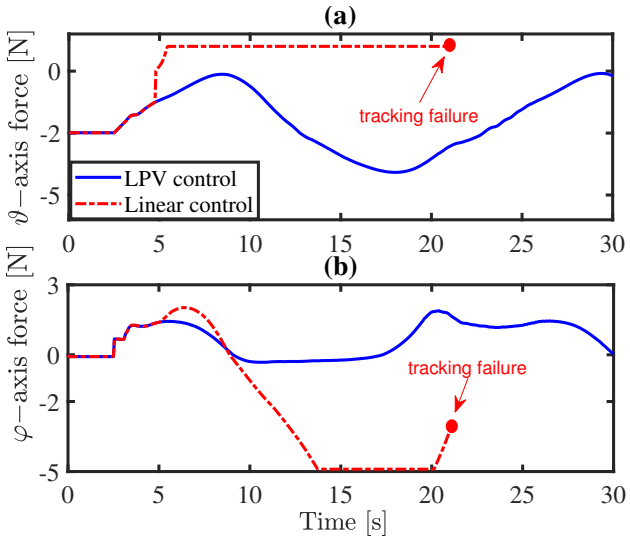


Figure 3.12: Force control inputs along the $\vartheta-$axis and the $\varphi-$axis (Test 1).

Phase 1 and Phase 2. We can observe in Figure 3.15(c) that due to the presence of the additional load, the $\varphi-$axis force control input is also reduced accordingly. Since the extra load only affects the $\varphi-$axis, there is no change for the $\vartheta-$axis force input between Phase 1 and Phase 2. Videos of this experiment can be found at: https://bit.ly/3RMDgBO.

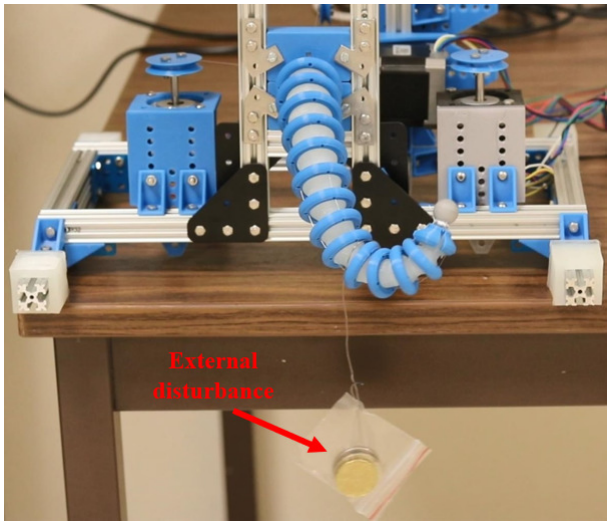


Figure 3.13: Soft Trunk robot with an additional load (Test 2).

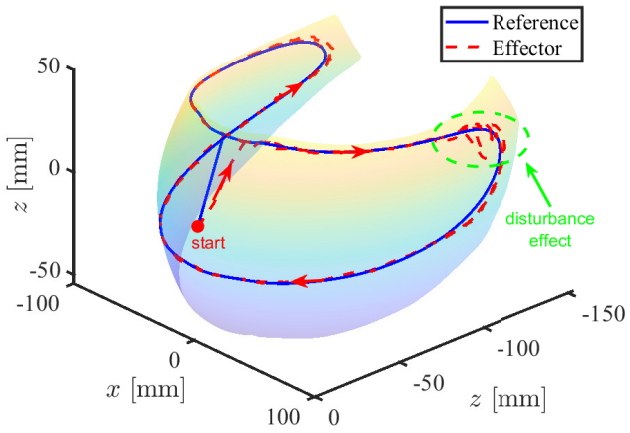


Figure 3.14: Experimental results of 3D dynamic tracking control in the presence of an external disturbance (Test 2).

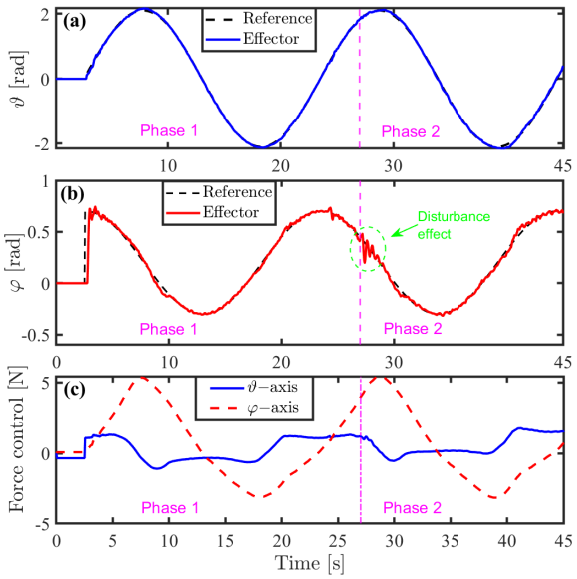


Figure 3.15: Tracking control task with an external disturbance in the parametric space (Test 2). (a) Tracking performance along $\vartheta-$axis. (b) Tracking performance along $\varphi-$axis. (c) Force control inputs.

Test 3: Comparisons with Iterative Learning Control

ILC aims at generating a feedforward control to track a reference trajectory of repetitive processes on a finite time interval while rejecting real-time disturbances [94]–[96]. For comparisons, we perform a circular trajectory tracking with the norm-optimal ILC control [97], which shares the common model-based principle as iterative learning model predictive control [98]. The designed ILC control scheme is composed of two decentralized single-input single-output (SISO) ILC controllers corresponding to the $\vartheta-$axis and the $\varphi-$axis, respectively. The design of each norm-optimal ILC controller is formulated as an optimization problem, as described in Appendix B. We distinguish two test scenarios for trajectory tracking: i) with a small robot deformation, and ii) with a large robot deformation.

a. Scenario 1: Tracking with a Small Deformation. This test scenario is performed with the following small-range reference such that linear control can be still effective:

$$\vartheta(t) = 0.4\sin(0.9t)\ [\text{rad}], \quad \varphi(t) = 0.4\cos(0.9t)\ [\text{rad}]. \tag{3.92}$$

The tracking control results obtained with both LPV control and ILC control are shown in Figure 3.16. We can see in Figures 3.16(a) and (c) that the tracking errors become smaller after each control iteration. After 15 iterations, the robot end-effector can track the desired reference (3.92) with the ILC control method. Moreover, the tracking control performance of both control methods is similar in this case.

b. Scenario 2: Tracking with a Large Deformation. For this test, the tracking control task is performed with the large-range reference defined in (3.91) to show that linear control is not effective anymore for large-deformation situations. The corresponding tracking control results are depicted in Figure 3.17. Observe that ILC control can improve loop-after-loop the tracking performance from the initial iterations, i.e., when the deformation is still small. However, when the deformation becomes large, the ILC controller is not effective anymore to track the desired reference. In particular, the end-effector is not able to reach the robot configuration with 90° bending deformation, which is not the case of the proposed LPV controller. Videos for both scenarios of this test can be found at: https://bit.ly/3xLb3CH.

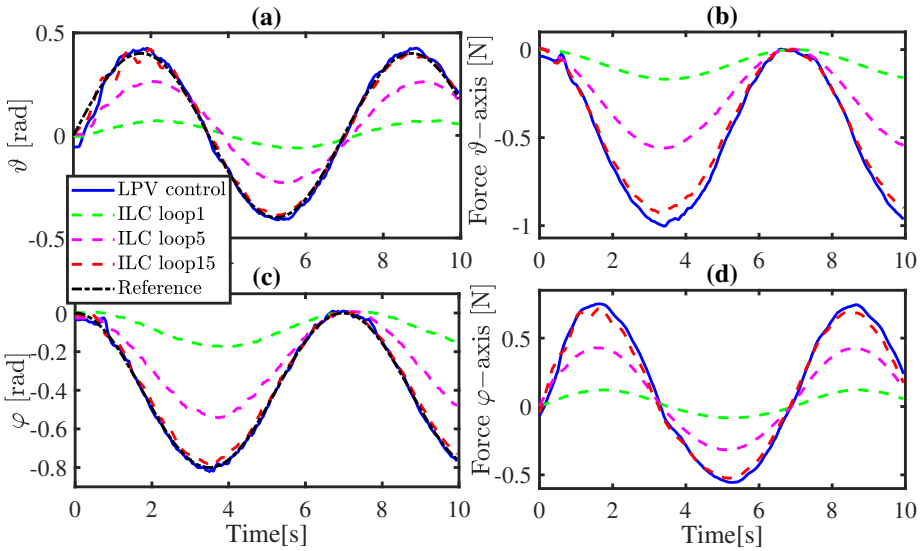


Figure 3.16: Tracking performance comparison between LPV control and different loops of ILC control for the small-deformation reference (3.92). (a) Tracking performance along $\vartheta-$axis. (b) Force control input along $\vartheta-$axis. (c) Tracking performance along $\varphi-$axis. (d) Force control input along $\varphi-$axis.

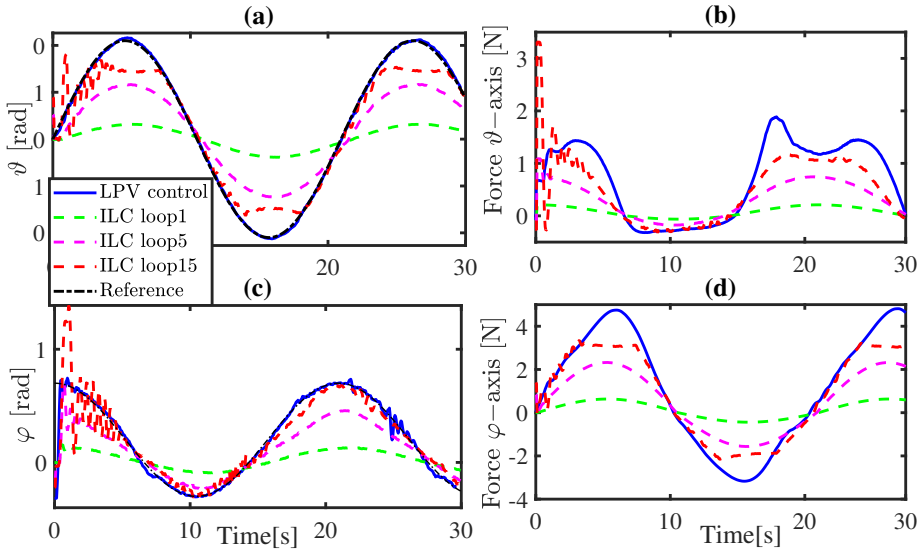


Figure 3.17: Tracking performance comparison between LPV control and different loops of ILC control for the large-deformation reference (3.91). (a) Tracking performance along $\vartheta-$axis. (b) Force control input along $\vartheta-$axis. (c) Tracking performance along $\varphi-$axis. (d) Force control input along $\varphi-$axis.

For a quantitative performance analysis, we define the following index of normalized square tracking error (nSTE) to avoid the impact of different reference amplitudes:

$$\text{nSTE} = \frac{1}{\|\boldsymbol{x}_r\|_{\ell_\infty}^2} \sum_{k=1}^{\Delta t} \|\boldsymbol{e}_k\|^2 \tag{3.93}$$

where Δt is the tracking time. Figure 3.18 summarizes the tracking performance in terms of nSTE index obtained with the proposed LPV control and the compared ILC controller for both small- and large-range trajectory references. Remark that for the small-range tracking, a clear performance improvement can be observed loop after loop for ILC control until a high tracking accuracy can be achieved. However, for the large-range tracking, despite a loop-after-loop improvement, the nSTE values obtained with ILC controller are large compared to that obtained with the proposed LPV controller, which gives similar nSTE values for both test scenarios. These confirm the tracking control results in Figures 3.16 and 3.17.

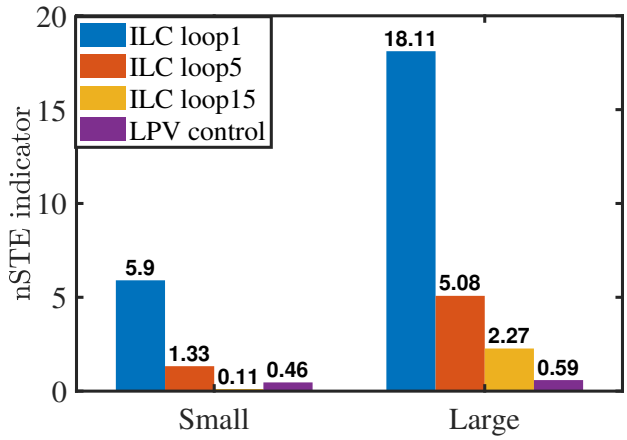


Figure 3.18: Comparisons between LPV controller and ILC controller in terms of nSTE performance for both small and large deformation test scenarios.

Test 4: Real-Time Marker Tracking

To further explore the potential of the proposed LPV control method, a more challenging experiment is conducted. The task is to imitate the elephant trunk to follow a moving target, i.e., to minimize the distance between the end-effector and the target, which is materialized by a marker as illustrated in Figure 3.19. Note that

the marker can be manually and arbitrarily moved within the workspace of the soft Trunk robot.

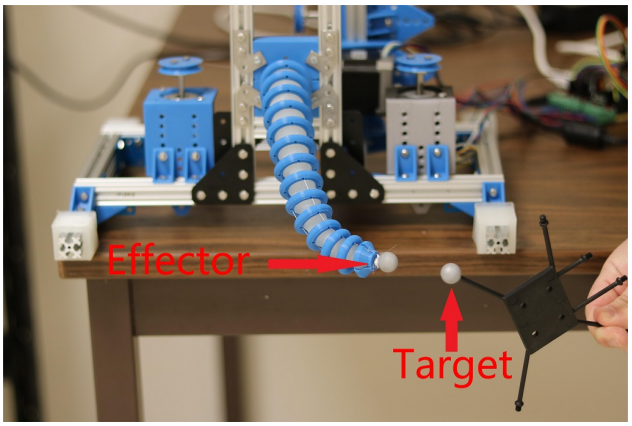


Figure 3.19: Illustration of a real-time target tracking task with the Trunk robot.

The 3D trajectories of the end-effector and the target are shown in Figure 3.20. For this test, we pay a special attention on the highly deformed configuration of the Trunk robot and the "unknown" and fast time-varying features of the target trajectory, which makes the tracking task much more challenging compared to the previous tests. Figure 3.21 shows that the real-time marker tracking is successfully achieved with the proposed LPV control method. Videos of this experiment can be found at: https://bit.ly/3KMDIOr.

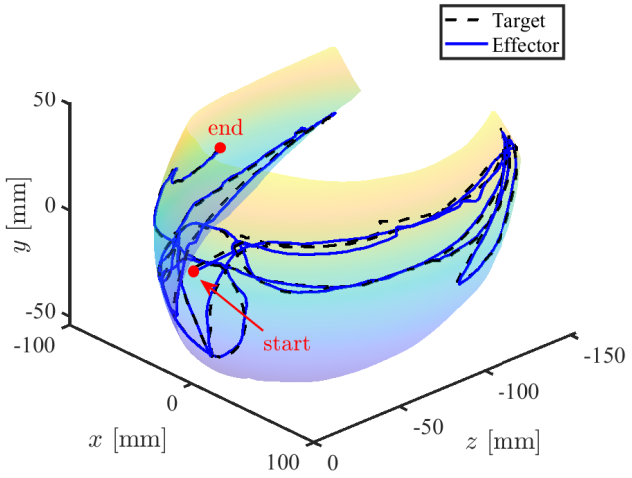


Figure 3.20: A 3D view of the real-time marker tracking task (Test 4).

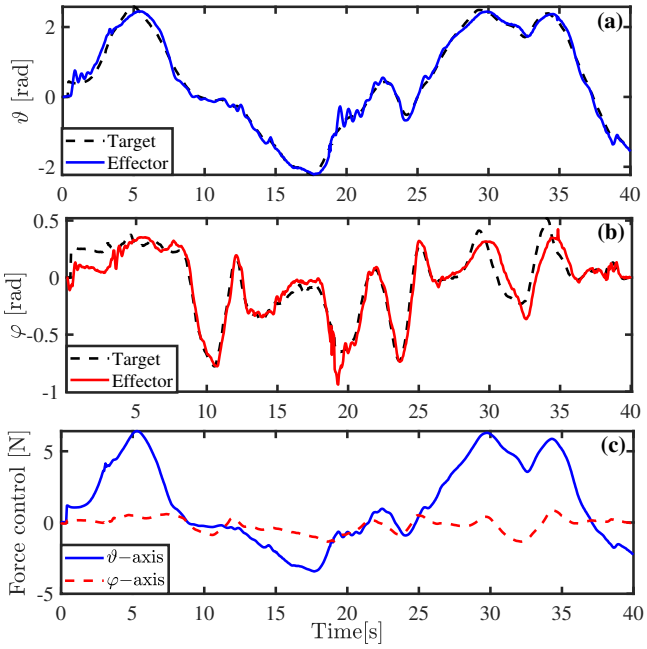


Figure 3.21: Real-time marker tracking task in the parametric space (Test 4). (a) Tracking performance along $\vartheta-$axis. (b) Tracking performance along $\varphi-$axis. (c) Force control inputs.

3.6 Concluding Remarks

A dynamic FEM model-based framework has been proposed for LPV tracking control of soft robots. Based on a POD model reduction method, we first generate a set of reduced-order linear models with the same mechanical structure for different operating points, covering the whole robot workspace. Using RBF networks, these local linearized models are interpolated to build a reduced-order LPV model, which can capture the nonlinear dynamics of soft robots with large deformations. Then, an EID-based control scheme is developed for LPV dynamic tracking control, which is composed of three key components, i.e., feedforward control, feedback control, error-compensation control. The LPV feedforward control is constructed from an interpolation of local control input values, obtained at the operating points considered for LPV modeling. The feedback control and the error-compensation control are designed with a generalized proportional integral observer structure, incorporating the low-frequency information of external disturbances and/or modeling uncertainties. Using Lyapunov stability theory, sufficient LMI conditions are derived to design both

the LPV feedback controller and the LPV extended observer such that the closed-loop soft robot system is globally uniformly $\ell_\infty-$stable. Various experimental tests have been carried out with a soft Trunk robot under configurations with both small and large deformations to validate the proposed LPV modeling and to show the effectiveness of the proposed LPV control method over existing linear tracking control results.

Appendix.B:Norm-Optimal ILC Controller Design

To design two decentralized ILC controllers, the reduced-order robot model (3.11) is decoupled into two SISO models corresponding to the control along the $\vartheta-$axis and the $\varphi-$axis, respectively. The norm-optimal ILC design for these both controllers follows the same procedure, which is summarized hereafter with the same formulation. More related technical details can be found in [99].

Under zero initial condition, the following impulse response matrix is derived from the linear robot model (3.11):

$$\mathbf{P} = \begin{bmatrix} C_p B_{rp} & 0 & \cdots & 0 \\ C_p A_r B_{rp} & C_p B_{rp} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_p A_r^{T-1} B_{rp} & C_p A_r^{T-2} B_{rp} & \cdots & C_p B_{rp} \end{bmatrix} \tag{3.94}$$

where T is the iteration length, i.e., $k \in [0, T]$. The pth input-output matrix pair (B_{rp}, C_p) corresponds to the SISO control loop along the $\vartheta-$axis or the $\varphi-$axis. The system output of the ith iteration can be represented by

$$\boldsymbol{y}_{k(i)} = \mathbf{P}\boldsymbol{u}_{k(i)} \tag{3.95}$$

where $\boldsymbol{u}_{k(i)}$ is the control input of the ith iteration. The norm-optimal ILC control design can be reformulated as the following optimization problem [99]:

$$\min_{\boldsymbol{u}_{k(i+1)}} \boldsymbol{e}_{k(i+1)}^\top W_e \boldsymbol{e}_{k(i+1)} + u_{k(i+1)}^\top W_u \boldsymbol{u}_{k(i+1)} \tag{3.96}$$

$$\text{subject to } \left[\boldsymbol{u}_{k(i+1)} - \boldsymbol{u}_{k(i)}\right]^\top \left[\boldsymbol{u}_{k(i+1)} - \boldsymbol{u}_{k(i)}\right] \leq \delta \tag{3.97}$$

where $\boldsymbol{e}_{k(i)} = \boldsymbol{r}_k - \boldsymbol{y}_{k(i)}$, and $\boldsymbol{r}_k$ is the output reference. The weighting matrices W_e and W_u determine the tradeoff between performance and input energy. The constraint (3.97) on mmaximum of control update can be taken into account in the

optimization problem (3.96) via a Lagrange multiplier λ as

$$\min_{\boldsymbol{u}_{k(i+1)}} \mathscr{J}_{k(i+1)} \tag{3.98}$$

with

$$\begin{aligned}\mathscr{J}_{k(i+1)} = \boldsymbol{e}_{k(i+1)}^{\top} W_e \boldsymbol{e}_{k(i+1)} + \boldsymbol{u}_{k(i+1)}^{\top} W_u \boldsymbol{u}_{k(i+1)} \\ + \lambda \left[\left(\boldsymbol{u}_{k(i+1)} - \boldsymbol{u}_{k(i)} \right)^{\top} \left(\boldsymbol{u}_{k(i+1)} - \boldsymbol{u}_{k(i)} \right) - \delta \right].\end{aligned}$$

Then, the optimal solution of the optimization problem (3.98) can be determined from the equation $\frac{\partial \mathscr{J}_{k(i+1)}}{\partial \boldsymbol{u}_{k(i+1)}} = 0$, leading to

$$\boldsymbol{u}_{k(i+1)} = \mathbf{W}_{\text{opt}}^{-1} \left(\lambda \boldsymbol{u}_{k(i)} + \mathbf{P}^{\top} W_e (\mathbf{I} - \mathbf{P}) \boldsymbol{r}_k \right) \tag{3.99}$$

with $\mathbf{W}_{\text{opt}} = \lambda \mathbf{I} + \mathbf{P}^{\top} W_e \mathbf{P} + W_u$. Substituting (3.95) into (3.99), the update law of the control input at the ith iteration can be obtained as

$$\boldsymbol{u}_{k(i+1)} = \mathbf{Q} \left[\boldsymbol{u}_{k(i)} + \mathbf{L} e_{k(i)} \right] \tag{3.100}$$

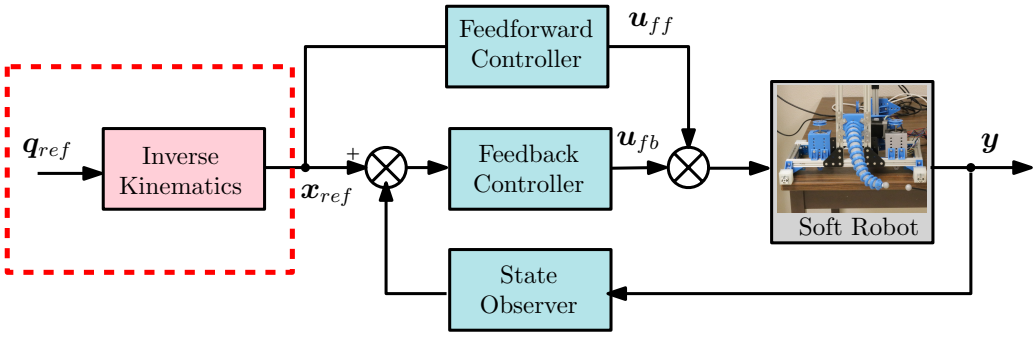
where the filter matrices $\mathbf{Q}$ and $\mathbf{L}$ are defined as

$$\begin{aligned}\mathbf{Q} &= \mathbf{W}_{\text{opt}}^{-1} \left(\lambda \mathbf{I} + \mathbf{P}^{\top} W_e \mathbf{P} \right) \\ \mathbf{L} &= \left(\lambda \mathbf{I} + \mathbf{P}^{\top} W_e \mathbf{P} \right)^{-1} \mathbf{P}^{\top} W_e.\end{aligned} \tag{3.101}$$

Remark 14. For ILC tuning, we select $W_e = I$ and $W_u = \rho I$, with $\rho \in [0, 1]$, to limit the computational burden of the control update law (3.100)–(3.101). The tuning of the two design parameters λ and ρ can be done as follows. Since the soft robot dynamics cannot be accurately described with a linear model, the value of λ should be large enough to maintain the system stability by avoiding an aggressive update in (3.100). The value of ρ should be small to minimize the tracking error $\boldsymbol{e}_{k(i)}$. For the experimental results in Section 3.5, we select $\lambda = 1.2$ and $\rho = 0.1$ for both SISO norm-optimal ILC controllers.

Chapter 4

Posture Based Inverse Kinematics for Soft Manipulators



After implementing a full-workspace nonlinear controller, arbitrary trajectories in workspace can be tracked. But if the robot is used to perform more advanced tasks such as touching or grasping, we need to know the corresponding target configuration and trajectory, which is known as motion or trajectory planning task. For a highly nonlinear control system such as a soft robot, a feedback controller is not enough to complete high level tasks, because its actual workspace is not a linear, but a manifold with special structure. This is actually the same problem that rigid robots face, but unlike rigid robots, soft robots inverse kinematic problem cannot be solved accurately in real-time.

Inspired by the modeling process in previous chapter, a novel kinematic representation of soft robot combining both model and collected data is proposed, as well as an inverse kinematics method based on measurement data.

4.1 Introduction

In this chapter, a learning and synthesis algorithm is presented for solving the IK problem of soft manipulator. The algorithm learns feasible configurations from collected configuration set and synthesizes new configurations that are similar to the learnt ones and satisfies the kinematic constraints. The real configurations $\{\boldsymbol{q}_i\}$, which consist of the coordinates of several points and end effector of the manipulator are collected by real-time experiment. The learning part is about training a model that connects high-dimension configurations with reduced order states, which can also be called feature vectors in machine learning. The feature vectors that we used to learn the configurations are obtained from the FEM model of soft manipulator as discussed in previous chapters. More specifically, our approach consists of following key points:

Generalized coordinates In order to have effective feature vector to represent a large number of soft manipulator postures, while maintaining the coherence with dynamical controller. We obtain the reduced order general coordinates $\boldsymbol{z}$ from large scale states $\boldsymbol{x}$ of FEM model by order reduction. The feature $\boldsymbol{z}$ is the same as the reduced order state vector $\boldsymbol{x}_r$ in previous chapters and can be reconstructed by state observer from actual measurement $\boldsymbol{y}$.

GPR learning To describe the likelihood between synthesized and captured postures of soft manipulator, the Gaussian Process Regressor (GPR) is introduced. The feature vector $\boldsymbol{z}$ and the captured posture $\boldsymbol{q}_i$ are the input and output of GPR respectively. A low dimentional representation of posture set is defined: each posture $\boldsymbol{q}_i$ has a corresponding feature $\boldsymbol{z}_i$, and the similarity between different postures is defined as the distance of their feature vectors.

IK solution To synthesize the desired posture of manipulator that both satisfy the constraints and is similar to collected postures, we formulate the IK as an optimization problem with an objective function $L_{IK}(\boldsymbol{z}, \boldsymbol{q})$ derived from the learnt GPR model.

4.2 Background and motivation

The main tasks of soft and rigid robots are actually highly overlapping, including reaching and moving given objects as well as avoiding obstacles by changing its

configuration. As an extension of rigid-body robots, soft-body robots also inherit the same problem about kinematics. The generalized coordinates of a robot are defined on a manifold thus linearized feedback controller is not effective to clarify the relevant issues, we can start with a rigid body robot for comparative research.

4.2.1 Control of rigid manipulator

The dynamical model of rigid robot manipulators is conveniently described by Lagrange dynamics. Using the robot manipulator with n links for example and let the $(n \times 1)$-vector $\boldsymbol{q}$ be the generalized coordinates (joint angles or translations). The dynamical model of the robot manipulator is given with Lagrange's equation [100]:

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{4.1}$$

where $\mathbf{H}(\boldsymbol{q})$ is the $(n \times n)$ inertia matrix, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the vector of Coriolis and centrifugal forces, $\boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q})$ is the vector of gravity force, and $\boldsymbol{\tau}$ is the generalized force vector that we assume to be the control input.

The kinematic model based on the geometrical properties is used to compute the position of the end-effector from specified values for the joint parameters. The coordinates of the end effector are functions of the generalized coordinates and can be given with

$$\boldsymbol{q}_y = f_{FK}(\boldsymbol{q}), \tag{4.2}$$

which is also known as forward kinematics (FK).

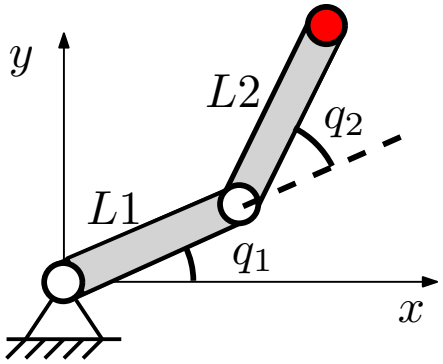


Figure 4.1: Kinematic model of a RR planar robot. The model has two joints angles as generalized coordinates.

From the viewpoint of controller design, end effector coordinates are the outputs of the system. The space of end effector is usually called task space and the space

of joint angles is called configuration space. Normally, the position, orientation and trajectory of the end effector are controlled to execute given tasks. To this end, joint angles are generated from desired trajectory of the end effector, and the problem is transformed into trajectory tracking of joint angles.

A baseline method for manipulator control is PD control with a gravity compensation scheme [100] of following form

$$\boldsymbol{\tau} = \mathbf{K}_{\mathrm{P}} \left(\boldsymbol{q}_{\mathrm{d}} - \boldsymbol{q} \right) - \mathbf{K}_{\mathrm{V}} \dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}), \tag{4.3}$$

where $\mathbf{K}_{\mathrm{P}}$ and $\mathbf{K}_{\mathrm{V}}$ are positive-definite gain matrices, $\boldsymbol{q}_{\mathrm{d}}$ is vector of desired joint angles. The closed-loop stability achieved by PD controller is analyzed according to the error dynamic based on (4.1). However, we may found it strange that the position of the end-effector is what we are concerned about but it is not appearing in the equation of the controller, so that it is not possible to describe the performance and robustness w.r.t system output. This compromise is due to the uniqueness of robot control problem.

Actually, as a non-linear dynamic model, holonomic constraints (geometrical relationships) have been embedded in the Lagrange's equation of rigid manipulator. That is to say, the method of motion planning can be directly carried out based on this equation, such as the trajectory planning problem of optimal time and energy, and the necessary conditions of the target trajectory are already provided by the optimal control theory [101]. While planning such trajectory is still an open and challenging problem, especially when simultaneously taking kinematic and dynamic constraints into account, which is called kinodynamic motion planning (KMP) [102]. These constraints are too complex for most robot platforms except for certain mobile and aerial robot, and it is hard to solve them analytically and even numerically.

In practice, the motion planning task is usually decoupled from dynamical model since the rigid manipulator can be precisely described with geometrical (kinematic) model.

4.2.2 Forward and inverse kinematics for rigid robot

Consider the rigid robot in Figure 4.1 for example, the forward kinematics describing the relationship from generalized coordinates $\boldsymbol{q} = [q_1, q_2]^T$ to end-effector position vector $\boldsymbol{q}_y$ is given with

$$\boldsymbol{q}_y = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) \\ L_1 \sin(q_1) + L_2 \sin(q_1 + q_2) \end{bmatrix}, \tag{4.4}$$

where q_1, q_2 are joints angles and L_1, L_2 are the lengths of links as shown in the figure. By substituting values of joint displacements into the right-hand side of the

kinematic equation, one can immediately find the corresponding end effector position and orientation.

For the motion planing task, we need to find the joint displacements that lead the end effector to the specified position and orientation. This is the inverse of previous problem, and it is called inverse kinematics. In terms of the planar robot used for example in Figure 4.1, the joint angles can be solved analytically. Given the position (x, y) of end effector, the corresponding joint angles can be calculated with

$$q_2 = \cos^{-1} \frac{x^2 + y^2 - a_1^2 - a_2^2}{2L_1 L_2}, \tag{4.5}$$

$$q_1 = \tan^{-1} \frac{y}{x} - \tan^{-1} \frac{L_2 \sin q_2}{L_1 + L_2 \cos q_2}. \tag{4.6}$$

while for robot with redundant structure, numerical methods are used in order to derive the desired joint displacements.

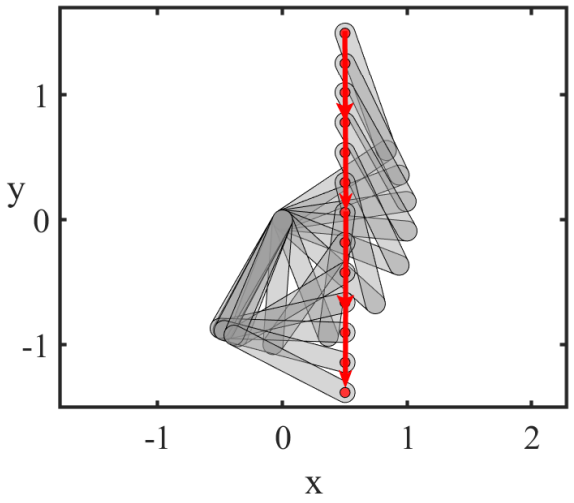
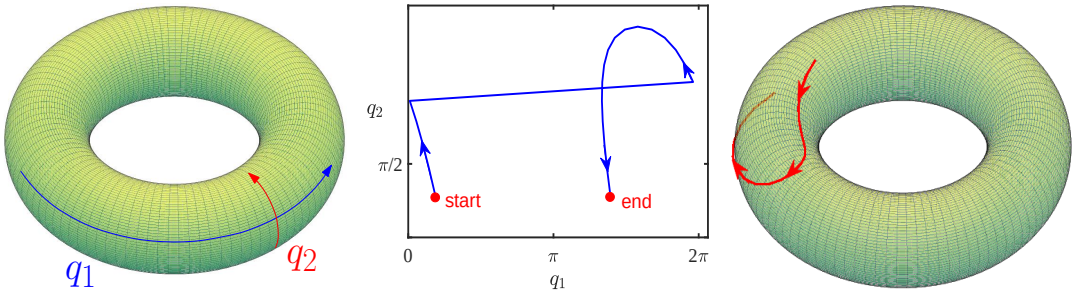


Figure 4.2: Configurations for vertical movement.

Figure 4.2 illustrated a series of manipulator configurations that move the end effector vertically. It can be seen that for a simple straight trajectory in the task space, the displacements of joints are complicated. Furthermore, since the dynamic and kinematic equations contain dozens of nonlinear trigonometric functions, the feasible region of linear state-space model of corresponding controller is highly restricted. A possible solution is to use LPV control framework based on nonlinear

dynamic model obtained from multibody dynamics [103] or identification from experimental data [104]. However, even with such non-linear controller, the inverse kinematics is still irreplaceable, because it is related to the topology of Configuration space(C-space).



(a) Topology of C-space (b) Linear space with trajectory (c) C-space with trajectory

Figure 4.3: The non-linear nature of robot C-space.

The planar robot in Figure 4.1 has two independent rotary joints, corresponding to two cyclic coordinates. Its configuration space is in the shape of a torus, as shown in Figure 4.3(a). Figure 4.3(b) shows a linear space of joint angles without special topological properties. The blue curve corresponds to the joint angle when the end effector of robot in Figure 4.2 moves along a vertical line. Since the topological structure is missed, the joint angles have to change discontinuously on axis q_1. The curves in Figure 4.3(c) depict joint angle trajectories on the manifold for the same trajectory.

In addition, since the process of linearization also leads to the loss of topology, the state trajectories in linear space actually do not fit the geometrical constraints of the manifold, which leads to a serious model mismatch. Therefore, the linear model, and its extension LPV model, cannot work alone when controlling objects such as robots. We need to use geometric (kinematic) information to obtain the desired trajectory and transform into a trajectory tracking problem.

4.2.3 Similar issue in soft robotics and related work

Soft manipulators have flexible structure and the capability to deform its configurations, which we would like to call it **posture** because we are concerned about the form of entire robot body instead of only the end effector. While during the manipulation of soft robots, we still need to follow the kinematics and geometrical

structures of general coordinates to perform control. Taking the robot in Figure 4.4 as an example, it is driven by cables to realize deformation along different directions. The robot can only bend but not stretch. During our transition of bending from left to the right, if we only design a feedback controller in the task space without considering the kinematics, then the desired trajectory is supposed to be the red one, which does not conform to the robot structure. One of the possible path is the curve shown in blue. It can be seen from the figure that a linear model is sufficient for a small range of bending, and by introducing the LPV model we can perform feedback control throughout the bending process. However, a kinematic model is indispensable to obtain the reference trajectory for LPV controller.

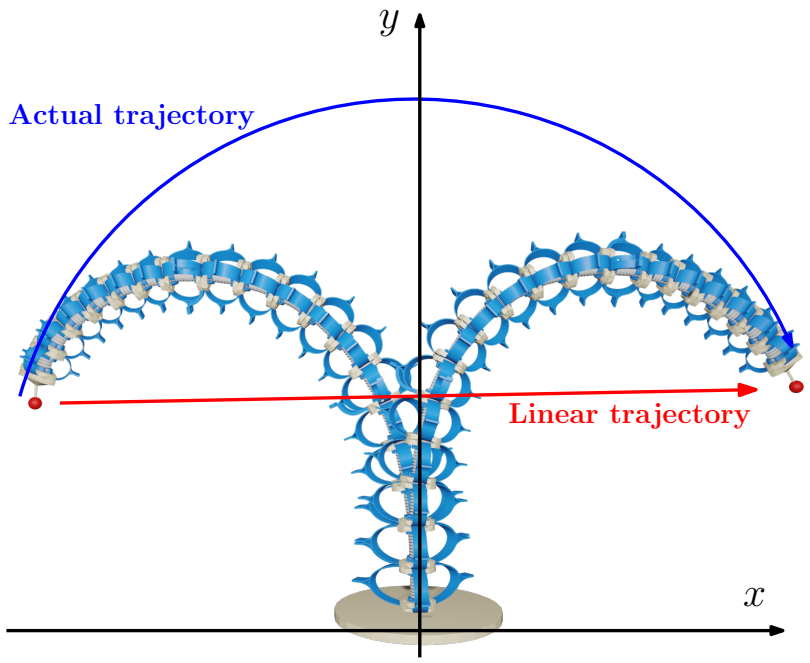
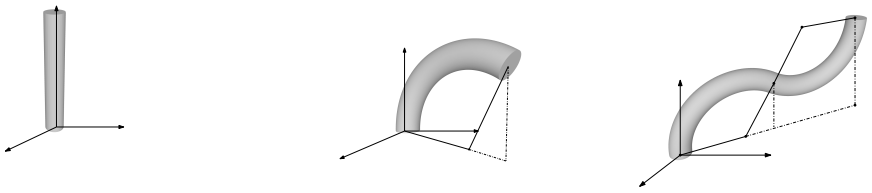


Figure 4.4: Illustration of bending deformation for Echelon robot [15].

4.2.4 Related works

Since the bending section of a continuum robot is in the form of an arc, a natural idea is to use multiple arcs to approximate the robot. This method of approximating robot with constant curvature has been applied in many studies. It is actually a replica of the rigid robot model and idealizes the deformation and geometry of soft body. When certain assumptions are fulfilled, the kinematic model of soft manipu-

lators can be formulated in an analytical way. In the literature, a typical case is the piece-wise constant curvature model [16].



(a) Original beam (b) Constant curvature section (c) Piece-wise constant curvature model

Figure 4.5: Piece-wise constant curvature element and model.

As the robot can be treated as several sections of curve with constant parameters, the relationship between sections can be described by homogeneous transformations [16]. Based on this rigid body transformation, the algorithms of rigid robot can be directly transferred and used. Therefore, the position of the end point can be converted into the curvature of each segment, and the inverse kinematics solution can be obtained through inverse transformation. This gives soft robots the possibility to accomplish specific tasks [45]. However, the method for rigid manipulators is less effective for soft cases, the deformation introduced by gravity and the interaction with load are ignored thus leading to significant modeling error and limited flexibility. Besides, for both dynamic and kinematic models of rigid manipulator, they share the same general coordinates (joint angles, etc.) which guarantees the combined implementation of motion planning and dynamic control. While the curvature is not a ideal coordinate for dynamical model, the solution of inverse kinematics is not available to be used in dynamical control.

There is another popular method known as differential kinematics or Jacobian kinematics, which does not use the relationship between general coordinates and the end effector position but their derivative:

$$\Delta \boldsymbol{q}_y = J(\boldsymbol{q})\Delta \boldsymbol{q}, \tag{4.7}$$

$$J(\boldsymbol{q}) = \frac{\partial \boldsymbol{q}_y}{\partial \boldsymbol{q}} = \frac{\partial f_{FK}(\boldsymbol{q})}{\partial \boldsymbol{q}}. \tag{4.8}$$

The Jacobian can be obtained from the geometrical model [16], Finite element method (FEM) model [39] or from experiment and regression of collected data [19].

For a rigid robot with an accurate geometric model, this method can be used to obtain the trajectories of joint angles through a iterative computation process. Since the model is of high accuracy, such trajectories can be directly used as the reference in the control of rigid robot. That is to say, trajectory planning and dynamic control can be performed separately. However, for a soft robot without precise kinematics, this algorithm can only plan the trajectory within a small range, and then the robot must update the model in reality and adapt the new configuration after making movement as shown in Figure 4.6. After the update process, the trajectory and control are re-planned and the accomplishment of task may take considerable iterations as shown in [15].

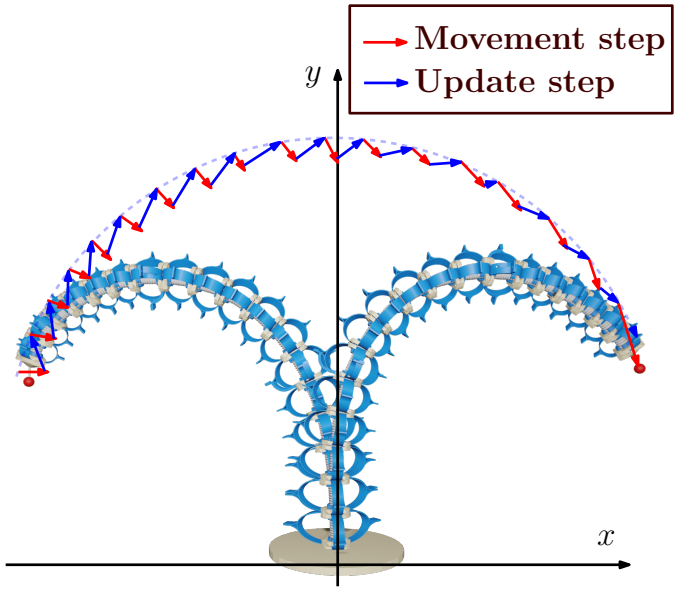


Figure 4.6: Movement process with Jacobian based method. The task is to move the end effector from left side to right side.

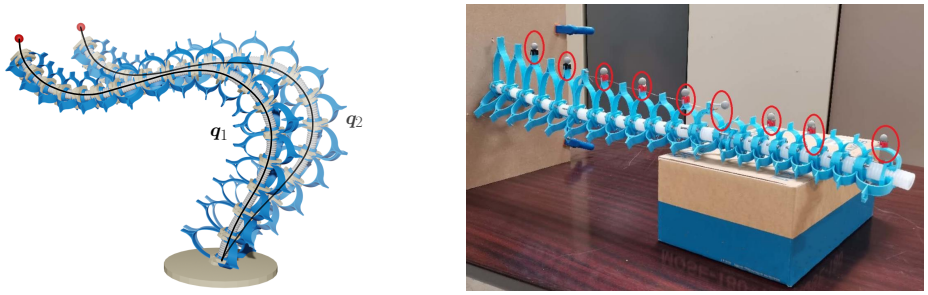
Therefore, such control scheme is an iterative loop of planning, execution and update. Here the available movement distance of each step is proportional to the accuracy of model. To guarantee the stability and smoothness, the end effector usually moves shortly and slowly at each step. Typical method from rigid robotics like is also introduced to overcome the convergence and singularity problems [53]. While obtain IK solution is still local, so that may be inappropriate for the global trajectory planning.

Due to the uncertain and non-linear nature of soft robots, data-based approaches

have also naturally been introduced. Due to the development and performance improvement of machine learning methods in the recent period, the two main mapping relationships in soft robots (forward kinematics [52], [105] and differential kinematics [52]) can be learned by machine learning models. After the training of regression model, the IK is formulated as a non-linear optimization problem. The required input for desired end-effector position can be obtained. After implementing obtained input as open loop control, local control is accomplished through Jacobian based method. This is an effective approach, but existing methods can only deal with the end-effector position instead of configuration whole robot, as well as corresponding local Jacobian method for feedback control. In order to realize the inverse kinematics solution and feedback control for the configuration of soft robot, we need to have deeper understanding about robot configurations and developing suitable method for representation and bridge the robot model with dynamic controller.

4.3 Postures and generalized coordinates

Among all the research about soft robotics, the flexibility of soft robots is always emphasized. But in terms of control and modeling, only simplified models are used, which is obviously contrary to our original intention. In order to obtain more information about soft manipulators and take full advantage of their flexibility, we must conduct more intensive sensing and measurement in soft robots. The stable structure of rigid-body robots enables us to characterize the robot's posture with only measured joint angles. While for soft robots, we need more measurement methods to capture robot configuration and other information. Motion capture systems, fiber-optic sensing [106] and image-based sensing [107] are supposed to be applied. The goal is no longer only the position of end effector, or the endpoint of each segment, but the curve of the whole robot body, so that the robot's flexibility and high degrees of freedom can be fully captured.



(a) (b)

Figure 4.7: Illustration of 3D model and real three-section Echelon robot [108]. Two different postures of robot are shown in (a). The robot has three sections with independent actuation, but the deformation is distributed in a more complicated way due to the robot structure and gravity. To capture the form of robot curve, in (b) we need at least 9 measured points. The optical markers are emphasized with red circles.

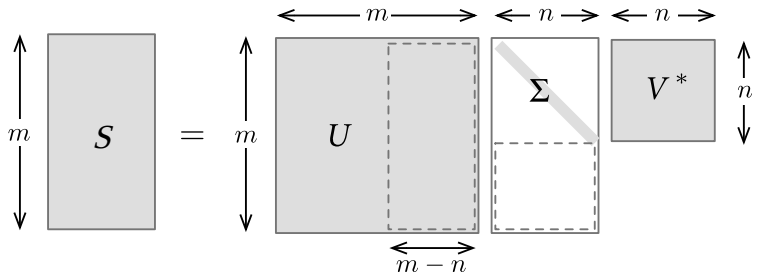
Another essential point is that the kinematics must be combined with the dynamic model. As mentioned earlier, the kinematic model must share generalized coordinates with the dynamic model, so that we can develop a corresponding controller and guarantee the realization of planned trajectory and reach the target configuration.

4.3.1 Another Perspective on POD

In the previous chapters, we first used FEM to model the soft robot, and then collected the different robot configurations as snapshots to reduce the model order. For the case that robot has state variable $\boldsymbol{q} \in \mathbb{R}^{m\times 1}$, the snapshot matrix $S \in \mathbb{R}^{m\times n}$ with n various configuration is given with:

$$S = (\boldsymbol{q}_{u_1}, \boldsymbol{q}_{u_2}, \ldots, \boldsymbol{q}_{u_n}). \tag{4.9}$$

The snapshot here includes robot configurations under various control inputs $\boldsymbol{u}_i$. Then, the projection matrix of POD is obtained by SVD of snapshot matrix:



The projector T of model order reduction from equation (2.8) is the truncation of matrix U and reduced order state variable $\boldsymbol{z} \in \mathbb{R}^{p\times 1}$ is given with

$$\boldsymbol{z} = T\boldsymbol{q}. \tag{4.10}$$

The projector T is used in previous chapters to obtain the reduced order dynamic systems, which enable us to design the feedback controllers.

POD can also be understood as the modal decomposition of snapshot. Here U is the orthonormal characteristic modal matrix and its column vectors are the modes arranged in descending order of singular values. The singular values in Σ are the coefficients of these modes to represent their energy. And the values in matrix V^* are the weights of different modes for every collected configurations. After completing the POD, all configurations of the robot become superposition of linearly independent modes of various orders.

We have shown earlier that the reduced-order general coordinates $\boldsymbol{z}$ is able to act as the state of dynamic system, based on which we can design feedback controllers, and here we can also see the potential of $\boldsymbol{z}$ to be the vector of general coordinates to represent various robot configurations like the joint angles of rigid manipulator.

4.3.2 Validation of configuration representation

Before introducing proposed inverse kinematic algorithm, we would like to validate the theory above with a simulation model to show the mechanism and performance. A planar soft robot is proposed because two dimentional figure is easier to visualize.

The configurations are generated through a piece-wise constant curvature model of a soft robot, which consists of two sections. The length of both sections are 20 [cm]. The model is actually a curve discretized with $m = 20$ states like in Figure 4.8(a).

First, we generate $n = 400$ different configurations with randomly assigned curvature of each section as shown in Figure 4.8(b), the interval of curvature is

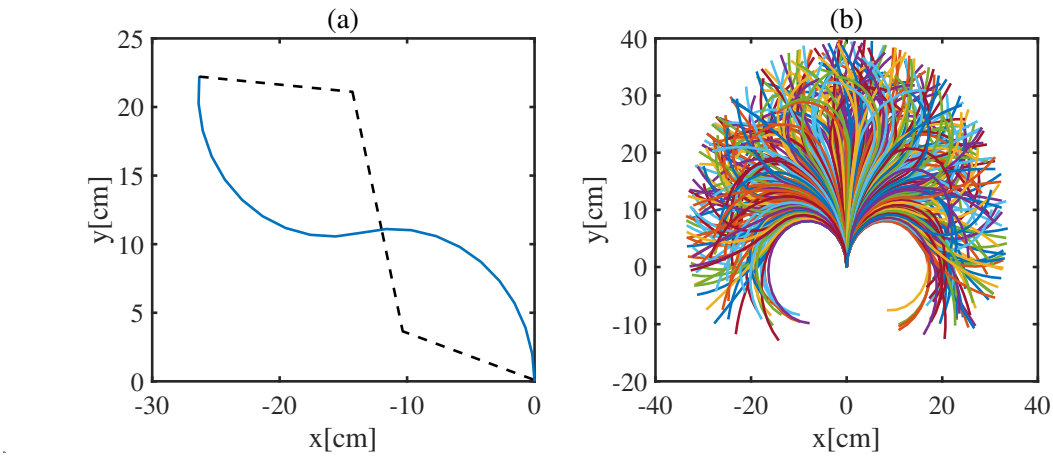


Figure 4.8: Simulated 2D postures. In (a), an example from dataset is shown, the model contains two PCC part I and II as illustrated, where the dotted lines are the radius of corresponding arcs. The data set contains 400 various configurations as shown in (b).

$[0.12, -0.12]$. Following the definition of snapshot matrix (4.9), the configuration vectors $\boldsymbol{q}_i$ are stacked to construct the snapshot matrix. To decide the order of reduced states, we first perform SVD to snapshot matrix and analyze the results for modal analysis. The distribution of singular values is shown in Figure 4.9, which indicates that with a reduced order of 2, we are able to take 68.6% of singular values and with an order of 4, we can take around 85.9%. Thus, the first 4 modes are a suitable compromise between accuracy and complexity. As discussed earlier, configurations of robot are decomposed to several orthonormal modes, which is the column of projector T. The first 4 modes in modal matrix U are illustrated in Figure 4.10(a) to show how the configurations can be dissembled. The result is consistent with well-known examples in vibration theory [109]. Each mode shape is an independent and normalized displacement pattern which may be amplified and superimposed to create a resultant displacement pattern, as shown in Figure 4.10(b). Since the illustrated robot only has 2 sections with constant curvatures, higher order displacement patterns are not significant.

The process of constructing proper features is called feature engineering in the research of machine learning, the idea of which is to construct a new variable to better represent data and improve the performance of regression. It is exactly what we have done for the representation of robot postures. So far, we have clarified how the reduced-order state $\boldsymbol{z}$ is obtained and the physical meaning behind.

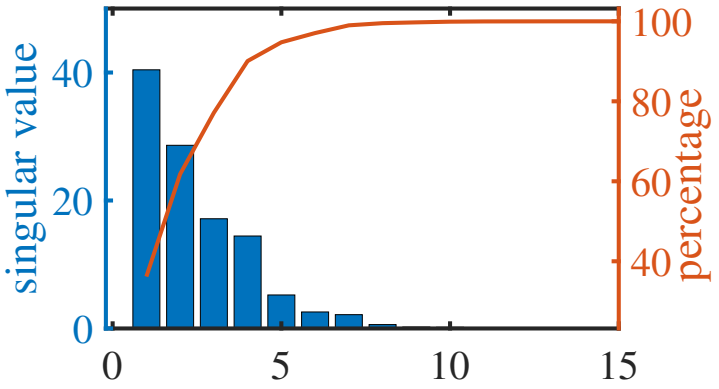
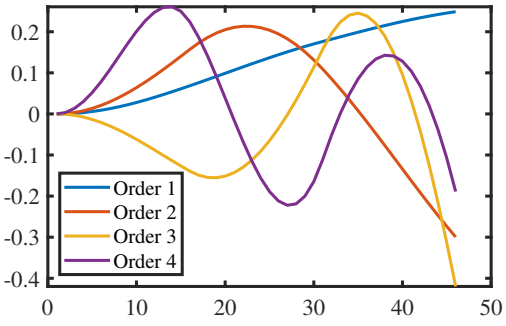
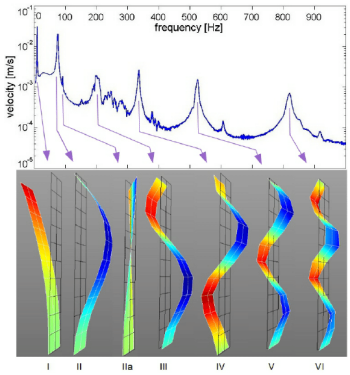


Figure 4.9: Left: the bar graph of singular values from the SVD decomposition of snapshot. Right: Accumulated percentage of singular values. Here, 4 order is enough to contain most of the modes because we have two section multiply with two dimension on x-axis and y-axis.



(a) Orthonormal modes from snapshot matrix of soft robot.



(b) Vibration modes of a cantilever beam.

Figure 4.10: Comparison of modal decomposition.

After obtaining the reduced-order feature vector $\boldsymbol{z}$ with equation (4.10), to visualize the distribution of postures w.r.t feature vector $\boldsymbol{z}$, these poses are illustrated using the coordinates of $\boldsymbol{z}$ in Figure 4.11. Here each blue point is corresponding to a robot configuration in the data set. It is worth noting that the postures automatically distributed in the form of arc, which is similar to the end effector positions in Figure 4.8.

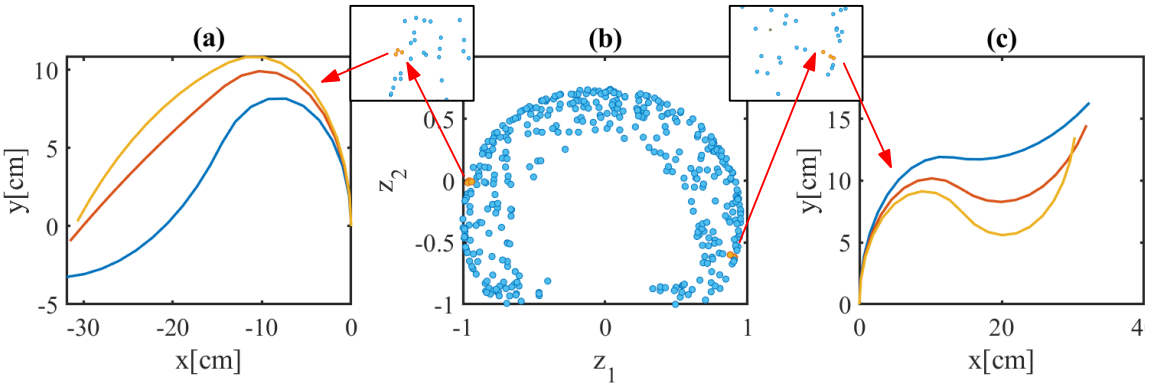


Figure 4.11: The comparison of configurations based on feature vector.

On the both sides of figure, postures in the interested regions are shown accordingly. We can observe that, in reduced-order feature space, the similar poses are nearby and the distribution is smooth. With the help of POD method, it is clear that the reduced state vector is a proper candidate for the feature vectors to represent various robot configurations. Based on this feature vector, we can develop the kinematic algorithms of soft robot.

4.4 Forward Kinematics (FK)

For rigid robots, the role of FK is to compute the position of the end-effector from given joint parameters. Since they are rigid body system, this relationship is purely geometric and a closed form can generally be obtained. In the soft robot case, this closed form cannot be obtained excepted for very simple geometries (a simple beam) and under assumptions (small deflection, Cosserat, ...). In addition, the advantages of soft robots are reflected in the flexible whole-body posture, while traditional kinematics only focuses on the position and orientation of the end effector. Here, we extend the definition of FK such that FK is the function to compute the posture curve from specified feature variables instead of only end effector.

The extended forward kinematics of soft manipulator can be defined as a vector-valued function as:

$$\boldsymbol{q} = \boldsymbol{f}(\boldsymbol{z}) \tag{4.11}$$

where $\boldsymbol{q} \in \mathbb{R}^{m\times 1}$ is robot posture and $\boldsymbol{z} \in \mathbb{R}^{p\times 1}$ is corresponding feature vector.

Due to the nature of soft manipulators, an analytical form of $\boldsymbol{f}(\boldsymbol{z})$ is not available and we cannot proceed with parameter identification method like rigid robot [100]. Instead, the FK function is able to be obtained from captured postures. We refer to non-parametric regression method that enables estimation of a continuous function from data by incorporating latent relationship between obtained posture and feature vectors.

4.4.1 Learning FK with Gaussian Process Model

Machine learning and in particular Gaussian process (GP) regression is suitable to estimate and predict unknown functions or models based on large amounts of data by introducing prior knowledge. With the available data and the efficiency of hardware implementations, GPs are gaining more attention in control recently, e.g., for MPC [110] and iterative learning control [111]. In the proposed method, Gaussian process plays an important role both in the forward kinematics and inverse kinematics because it is able to characterize prediction uncertainty which is essential for the following IK solution.

The realization of GP in this chapter is classical with vector output [112], but we will go through with some key steps in Bayesian analysis to show the special interest of the GP on FK learning problem. A detailed exposition can be found, for instance in [113]. In the training process, we have a data set with N observations, $\mathcal{D} = \{(\boldsymbol{z}_i, \boldsymbol{q}_i)|i = 1, \ldots, N\}$ collected from the motion capture system to initialize the regression model, as well as concatenated feature vector $\mathbf{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N]^T$ and concatenated posture vector $\mathbf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N]^T$.

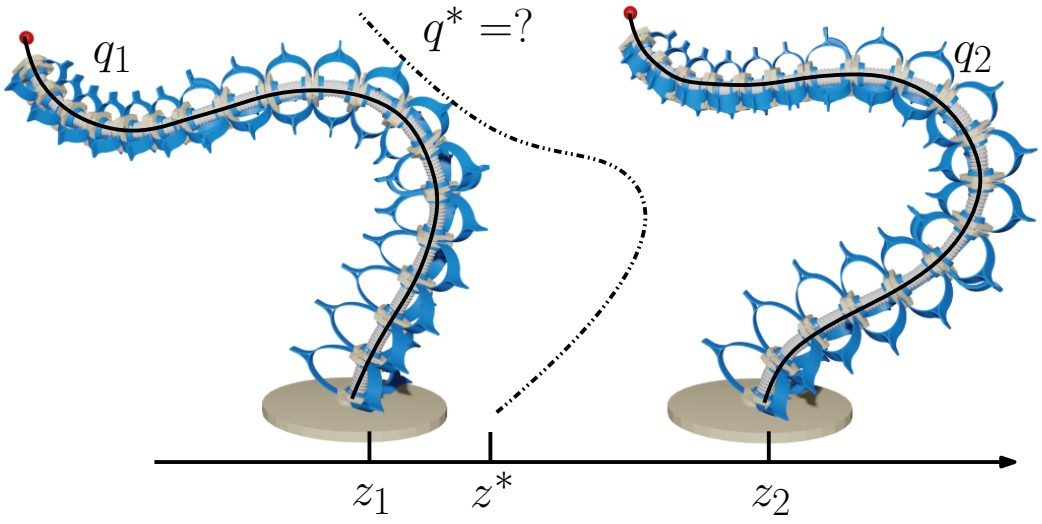


Figure 4.12: The FK problem is about constructing new postures $\boldsymbol{q}^*$ based on known poses given feature vector $\boldsymbol{z}^*$. The new posture is supposed to be more similar to posture q_1 since their feature vectors are closer.

The learnt FK model is supposed to construct new robot posture $\boldsymbol{q}^*$ for given feature vector $\boldsymbol{z}^*$, based on the known data set $\mathcal{D}$ as illustrated in Figure 4.12. As we said in last section, the FK function $\boldsymbol{f}(\boldsymbol{x})$ is continuous, the new posture can be generated from similar postures that are close to it. In Figure 4.12, new pose $\boldsymbol{q}^*$ will be interpolated with $\boldsymbol{q}_1, \boldsymbol{q}_2$, since $\boldsymbol{z}^*$ is closer to $\boldsymbol{z}_1$, the interpolation will be a weighted sum by taking into account the similarity.

Gaussian kernel function $k(\boldsymbol{z}, \boldsymbol{z}')$ acts as the measurement of similarity between postures $\boldsymbol{q}$ and $\boldsymbol{q}'$ based on the Euclidean distance of corresponding feature vectors $\boldsymbol{z}$ and $\boldsymbol{z}'$:

$$k(\boldsymbol{z}, \boldsymbol{z}') = \exp(-\beta \|\boldsymbol{z} - \boldsymbol{z}'\|^2) \tag{4.12}$$

where parameter β determines the spread of Gaussian function to be concentrated or smooth.

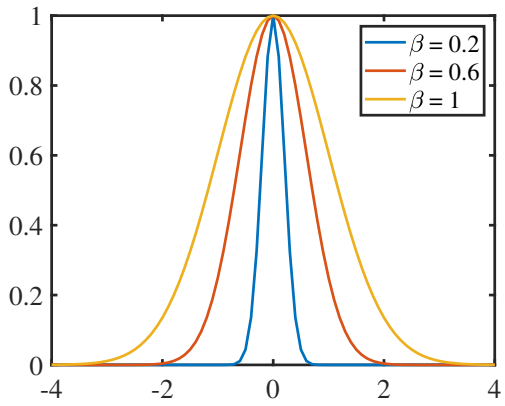


Figure 4.13: Gaussian Kernel with different spread β. Two given postures are similar when their feature vectors are close. A larger β means a larger credible interval.

The prediction $\hat{\boldsymbol{f}}(\boldsymbol{z})$ of the GPR is a weighted sum of prediction vector $\boldsymbol{w}$ and weights are kernel functions:

$$\hat{\boldsymbol{q}} = \hat{\boldsymbol{f}}(\boldsymbol{z}) = \boldsymbol{\phi}(\boldsymbol{z})^\top \boldsymbol{w}, \tag{4.13}$$

where

$$\boldsymbol{\phi}(\boldsymbol{z}) = (k(\boldsymbol{z}, \boldsymbol{z}_1), k(\boldsymbol{z}, \boldsymbol{z}_2), \ldots, k(\boldsymbol{z}, \boldsymbol{z}_N))^\top,$$

$\boldsymbol{w}$ is the parameter to be learnt. Here vector $\boldsymbol{\phi}(\boldsymbol{z})$ describes the similarity between interested feature $\boldsymbol{z}$ and each posture $\boldsymbol{z}_i$ in the data set. In $\boldsymbol{\phi}(\boldsymbol{z})$ the elements that are close to query value have larger values. From (4.11) and (4.13) we have the regression model

$$\boldsymbol{q} = \boldsymbol{\phi}(\boldsymbol{z})^\top \boldsymbol{w} + \boldsymbol{\varepsilon} \tag{4.14}$$

where $\boldsymbol{\varepsilon}$ is assumed to be i.i.d. Gaussian noise with zero mean and variance σ_n^2

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{\sigma}_n^2 I). \tag{4.15}$$

The noise origins from the reduction error and inaccuracy of the measurement by the motion capture system, each dimension of $\boldsymbol{\varepsilon}$ has independent and identically distributed probability distribution (i.i.d.). For the data set with N samples, we also define the $N \times N$ covariance matrix $K(\mathbf{Z}, \mathbf{Z})$, in which $K_{i,j} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$ and $\mathbf{k}(\boldsymbol{z}, \mathbf{Z}) = \phi(\boldsymbol{z})$ is one row in $K(\mathbf{Z}, \mathbf{Z})$. The covariance matrix defines the similarity between the elements in the data set.

From (4.14), we can obtain the likelihood function of all collected training prior set as multivariate Gaussian distribution [113]

$$\mathbf{Q} \sim \mathcal{N}(\mathbf{0}, K(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 I). \tag{4.16}$$

Then, the joint distribution of collected and estimated postures is given with [113]

$$\begin{bmatrix} \mathbf{Q} \\ \boldsymbol{q}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} K(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 I & \mathbf{k}\left(\mathbf{Z}, \boldsymbol{z}^*\right) \\ \mathbf{k}\left(\boldsymbol{z}^*, \mathbf{Z}\right) & k\left(\boldsymbol{z}^*, \boldsymbol{z}^*\right) \end{bmatrix} \right), \tag{4.17}$$

where the covariance between estimated posture and postures in data set is defined through $\mathbf{k}\left(\boldsymbol{z}^*, \mathbf{Z}\right)$.

The prediction of a new posture $\boldsymbol{q}^*$ is the posterior mean under known observation $\mathbf{Q}$ and input $\boldsymbol{z}^*$. It can be obtained by deriving conditional distribution [113] from (4.17) as

$$p(\boldsymbol{q}^* | \boldsymbol{z}^*, \mathbf{Q}, \mathbf{Z}) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}), \tag{4.18}$$

where

$$\hat{\boldsymbol{\mu}} = \mathbf{k}(\boldsymbol{z}^*, \mathbf{Z})(K(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 I)^{-1}\mathbf{Q} = \mathbf{k}(\boldsymbol{z}^*, \mathbf{Z})\boldsymbol{w}, \tag{4.19}$$

$$\hat{\boldsymbol{\sigma}} = k(\boldsymbol{z}^*, \boldsymbol{z}^*) - \mathbf{k}\left(\boldsymbol{z}^*, \mathbf{Z}\right)\left(K(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 I\right)^{-1}\mathbf{k}\left(\mathbf{Z}, \boldsymbol{z}^*\right). \tag{4.20}$$

Here $\boldsymbol{q}^* = \hat{\boldsymbol{\mu}}$ is the constructed posture of feature vector $\boldsymbol{z}^*$ with variance $\hat{\boldsymbol{\sigma}}$.

We can notice that the value of element in variance $\hat{\boldsymbol{\sigma}}$ only depends on the distance from known feature vectors $\boldsymbol{z}_i \in \mathbf{Z}$ to query vector $\boldsymbol{z}^*$. The variance (uncertainty) is greater when $\boldsymbol{z}^*$ is far from training data and we have less confidence.

Remark 15. We can pay attention to equation (4.19) to understand how the method is working. Each time when the posture corresponding to feature vector $\boldsymbol{z}^*$ is required, the similarity between required posture and collected postures are computed first as $\mathbf{k}(\boldsymbol{z}^*, \mathbf{Z})$. Since the postures are stored in matrix $\mathbf{Q}$, the predicted posture $\boldsymbol{q}^*$ is actually the weighted sum of postures in data set. And the variance $\hat{\boldsymbol{\sigma}}$ in (4.20) acts as the measurement of uncertainty to be used in next section.

4.5 Inverse Kinematics

Soft manipulators behavior are much more sensitive to fabrication error, external forces (load, contact, ...) and actuation performances, thus a IK method having same performance as rigid robot is impractical. From the observation of object reaching behavior of elephant trunk [20], we can find that the movement contains two stages:

a feedforward stage that helps elephant to move the trunk rapidly but coarsely to the target and a feedback stage to guarantee the accomplishment of task, which is same as human beings. Based on this, we define the target of IK for soft manipulator is to give an acceptable estimation of desired posture that fits with collected poses and satisfies given constraints. The problem of generating inverse kinematics solutions for interactive characters from data [114] and finite element models [115] are also involved in computer graphics, and the algorithms in this chapter are also inspired by these studies.

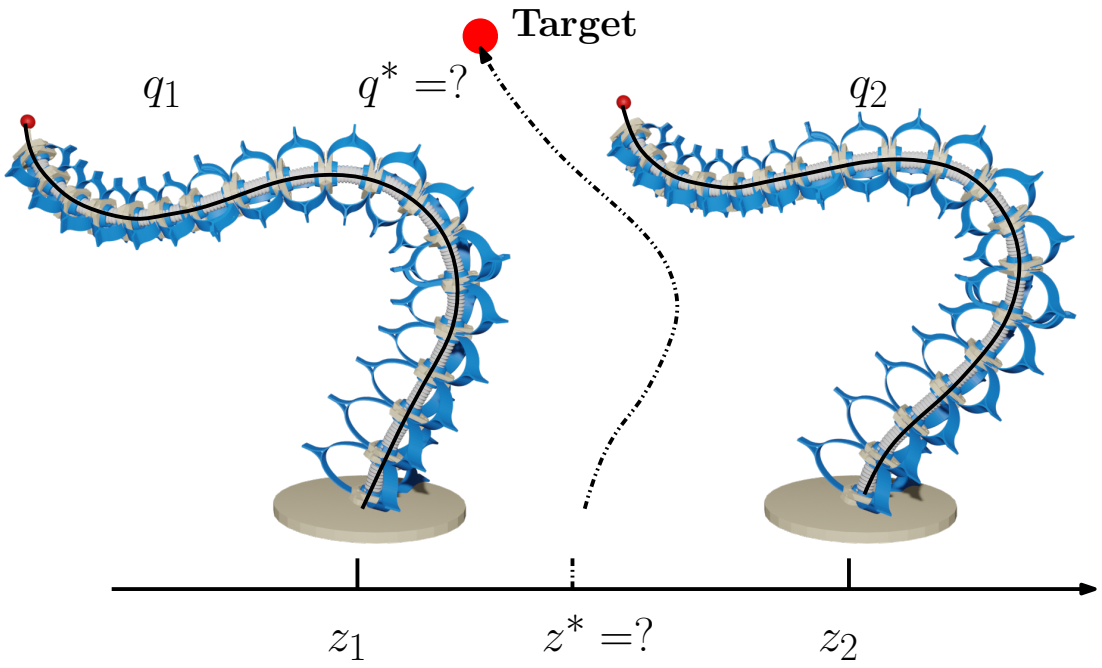


Figure 4.14: The IK problem is about constructing new postures $\boldsymbol{q}^*$ and feature vector $\boldsymbol{z}^*$ given collected poses and constraints.

In this section, a new method for solving the IK problem of soft manipulator is presented. The new IK method could generate robot postures that conform constraints and the new postures are similar with collected postures in data set. It uses the previously trained FK Gaussian Process model to evaluate the credibility of generated new postures. Unlike the classical IK method that using analytical geometrical models, proposed method is based on the sampled points of posture curve, hence it is more suitable for robot with large deformation and high nonlinear

structures.

4.5.1 Methodology

Using the same notation that $\boldsymbol{q} \in \mathbb{R}^{m\times 1}$ is robot postures and $\boldsymbol{z} \in \mathbb{R}^{p\times 1}$ is corresponding feature vector, the IK can be formulated as an optimization problem of objective function $L_{IK}(\boldsymbol{z}, \boldsymbol{q})$ derived from previous section:

$$\arg\min_{\boldsymbol{z},\boldsymbol{q}} L_{IK}(\boldsymbol{z}, \boldsymbol{q}) \tag{4.21}$$

$$\text{s.t.} \quad C(\boldsymbol{q}) = 0. \tag{4.22}$$

where

$$L_{IK}(\boldsymbol{z}, \boldsymbol{q}) = \frac{||\boldsymbol{q} - \hat{\boldsymbol{f}}(\boldsymbol{z})||^2}{2\hat{\boldsymbol{\sigma}}^2(\boldsymbol{z})} + \frac{n}{2}\ln\hat{\boldsymbol{\sigma}}^2(\boldsymbol{z}) + \frac{1}{2}||\boldsymbol{z}||^2. \tag{4.23}$$

here $C(\boldsymbol{q})$ is a set of constraints on the robot posture, $\hat{\boldsymbol{f}}(\boldsymbol{z})$ is the FK evaluation on feature vector $\boldsymbol{z}$ based on GPR in (4.19), $\hat{\boldsymbol{\sigma}}^2$ is the variance of conditional distribution of FK prediction in (4.20).

The objective function L_{IK} can be interpreted as follows. To generate a posture that is similar to collected data, we optimize the robot posture $\boldsymbol{q}$ and its feature vector $\boldsymbol{z}$ at the same time, and the first term in objective function $\frac{||\boldsymbol{q}-\hat{\boldsymbol{f}}(\boldsymbol{z})||^2}{2\hat{\boldsymbol{\sigma}}^2(\boldsymbol{z})}$ can be interpreted as soft constraints on postures, which try to keep optimized posture $\boldsymbol{q}$ similar to learnt distribution of postures $\hat{\boldsymbol{f}}(\boldsymbol{z})$ when the hard constraints $C(\boldsymbol{q})$ have to be satisfied. While the second term $\frac{n}{2}\ln\hat{\boldsymbol{\sigma}}^2(\boldsymbol{z})$ is designed to minimize the uncertainty of prediction by keeping the feature vector $\boldsymbol{z}$ close to collected set $\mathbf{Z}$ as shown in Figure 4.15, closer distance lead to smaller variance. The last part $||\boldsymbol{z}||^2$ is the regularization term to avoid overfitting.

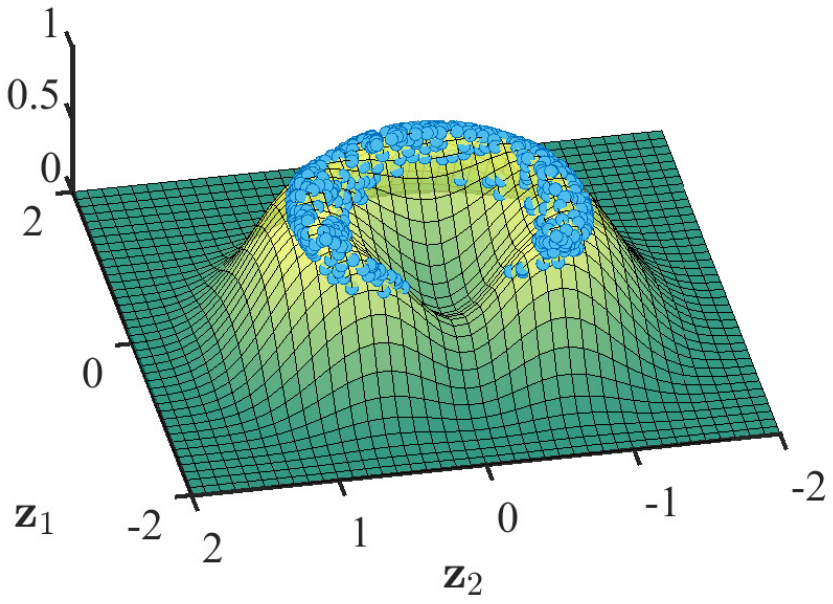
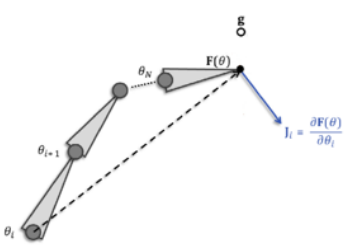
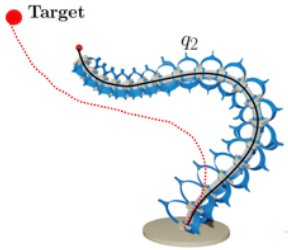


Figure 4.15: The credibility graph of 2D feature vector $\boldsymbol{z}$ from collected postures (blue point) in Figure 4.8. The second term of covariance $\hat{\sigma}$ in equation (4.20) is visualized as surface. The feature vector is trusted to be reliable around collected points in data set and credibility is positively related to data density. The term $\frac{n}{2}\ln\hat{\boldsymbol{\sigma}}^2(\boldsymbol{z})$ in objective function limits the new feature vector to be located in high credibility region, thus keeping it close with data set.

Remark 16. A simpler idea is to only optimize feature vector $\boldsymbol{z}$ and the estimated posture $\boldsymbol{q}$ is directly generated by trained GPR with (4.19), as shown in [19]. This follows the same manner of inverse kinematics for rigid robots, where the joint angles are variables to be optimized. While the constraints are designed for the robot posture $\boldsymbol{q}$ instead of optimization variable $\boldsymbol{z}$, this can not be formulated as a typical optimization problem.

The summarized method from FK to IK is shown in Algorithm.2. In order to provide a clearer perspective on the proposed IK method, we summarize the main elements of the method in the Table 4.16 and compare it with the IK of rigid-body robots.





Forward kinematics	analytical function $\mathbf{F}(\boldsymbol{\theta})$	Gaussian Process model $\hat{\boldsymbol{f}}(\boldsymbol{z})$
Output	end effecor position $\boldsymbol{y}$	robot posture $\hat{\boldsymbol{q}}$
Genralized coordinates	joint angle $\boldsymbol{\theta}$	reduced order states $\boldsymbol{z}$
Inversed kinematics	optimize $\min_{\theta} \frac{1}{2}\|\mathbf{g} - \mathbf{F}(\boldsymbol{\theta})\|^2$	optimize $L_{IK}(\boldsymbol{z}, \boldsymbol{q})$

Figure 4.16: Comparison of proposed IK method for soft robots and Jacobian based IK method for rigid robots.

4.6 Numerical Validation

4.6.1 Setup

The simulation to test our method is performed on the PCC model of a three-section planar soft manipulator. The length of each section is 20[cm] with maximum curvature as 0.25. Here, the robot has more degrees of freedom than the one used in previous section for illustration. An example of generated robot postures is shown in Figure 4.17.

Algorithm 2 Algorithm for inverse kinematics

Step 1 - Training of Forward Kinematics:

Inputs:

Data set of postures $\boldsymbol{q}_i$ and feature vector $\boldsymbol{z}_i$: $\mathcal{D} = \{(\boldsymbol{z}_i, \boldsymbol{q}_i) | i = 1, \ldots, N\}$

Outputs:

Trained Gaussian Process and output vector $\boldsymbol{w}$

Required parameters:

Kernel function $k(\boldsymbol{z}, \boldsymbol{z}')$ from (4.12)

Variance σ_n^2

Begin

Assemble posture matrix $\mathbf{Q}$ by stacking postures $\boldsymbol{q}_i$

Compute output vector $\boldsymbol{w} = (K(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2 I)^{-1}\mathbf{Q}$ with (4.19)

End

Step 2 - Inverse kinematic solution:

Inputs:

Desired position of end-effector or desired robot posture

Gaussian process parameter $\boldsymbol{w}$

Outputs:

Feature vector $\boldsymbol{z}$ for IK solution

Posture vector $\boldsymbol{q}$ for IK solution

Initialization:

Initialize vectors $\boldsymbol{q}$ and $\boldsymbol{z}$ with relaxed posture of robot

Begin

Formulate constraints on robot posture as $C(\boldsymbol{q}) = 0$

Minimize objective function $L_{IK}(\boldsymbol{z}, \boldsymbol{q})$

End

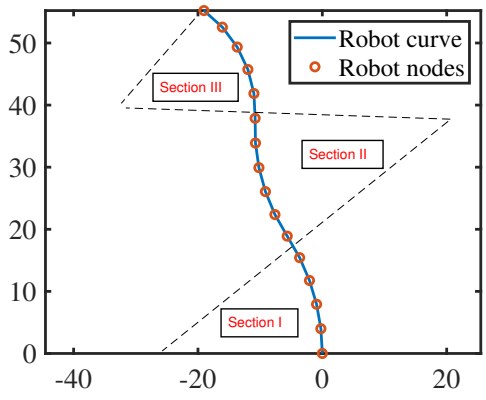


Figure 4.17: Illustration of a three-section planar robot generated with PCC model. The robot is discretized into 15 subsections.

The robot is redundant for the task in $x-y$ plane, and thus becoming more challenging for the IK solution. The robot is discretized with 15 points as posture vector $\boldsymbol{q}$. In the simulation, we design the kinematic constraints of the soft robot as the position of the end effector, which is also the most common robotic task.

The training data set $\mathcal{D}_T = \{\boldsymbol{q}_i, \boldsymbol{z}_i\}, i = 1, \ldots, N$ consists of $N = 5000$ pairs of feature vector and postures, where the postures are generated with random curvature for each section and the reduced feature vector $\boldsymbol{z} \in \mathbb{R}^{6\times 1}$ is obtained from the POD process and the order of feature vector is determined from the diagram of singular values in Figure 4.9. Besides, we also generated a validation data set $\mathcal{D}_V$ with 500 random postures to test trained FK model. In our practice, the data processing step like normalization and zero-centering is required to increase the performance of the model. We could also have mean value functions $\boldsymbol{\mu}(\mathbf{z})$ and $\boldsymbol{\mu}(\boldsymbol{q})$ become zero for simplicity. The hyper parameters of measurement noise variance σ_n and kernel function spread β are first obtained from maximum likelihood estimation [113] and has to be manually tuned.

4.6.2 FK Validation

The GP of FK is initialized on the training data set $\mathcal{D}_T$ and then validated on set $\mathcal{D}_V$. The proposed FK algorithm is actually a regressor, the performance of interpolation can be measured with the 2- norm of error vector $\|\boldsymbol{q} - \boldsymbol{q}^*\|_2$. During the test, we are comparing the postures $\boldsymbol{q}_i$ in $\mathcal{D}_V$ with the generated posture $\boldsymbol{q}_i^*$ by GP model from feature vector $\boldsymbol{z}_i$ of $\boldsymbol{q}_i$. The results of FK validation is shown in Figure 4.18.

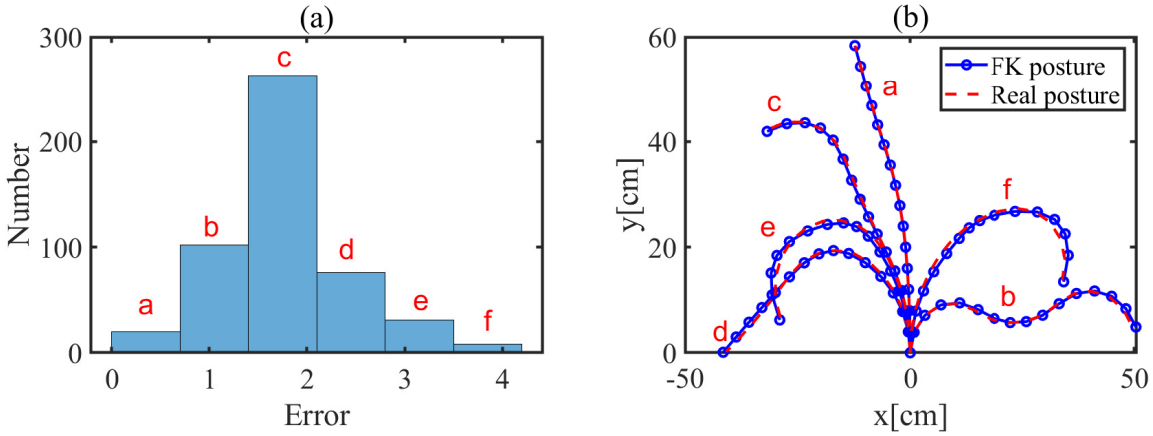


Figure 4.18: The distribution of interpolation error $\|\boldsymbol{q} - \boldsymbol{q}^*\|_2$ is illustrated in Figure(a), and 6 postures corresponding to different error class are presented in Figure(b).

Among all of 500 postures in the validation set $\mathcal{D}_V$, the interpolation error is varying from 0.283 to 3.831 and concentrated around 2. The performance of the algorithm is stable and satisfying on the tested data. Postures from each class in the histogram are provided to show the error visually. It can be observed from posture (f) that the interpolation error is satisfied even for the worst case. The reduction based FK regression can represent postures effectively and we will use learnt parameters of GP in the IK method.

4.6.3 IK Validation

The IK task in the validation is typical reaching movement of manipulator, the desire position of robot end effector is given and it is formulated as a constraint $C(\boldsymbol{q})$ of the optimization problem. Then the target function L_{IK} is optimized with Sequential quadratic programming (SQP) solver and gives the new posture $\boldsymbol{q}^*$ and its feature vector $\boldsymbol{z}^*$. As we discussed earlier, the generated postures are supposed to be similar with collected ones. To illustrate the effectiveness of proposed method, we also compare the postures reconstructed from feature vectors by reversing the POD mode order reduction process:

$$\text{Reconstruction} : \hat{\boldsymbol{q}} = T^T \boldsymbol{z}^*. \tag{4.24}$$

mThe reconstructed posture $\hat{\boldsymbol{q}}$ is the actual posture corresponding to feature vector $\boldsymbol{z}^*$. The posture $\boldsymbol{q}^*$ and feature vector $\boldsymbol{z}^*$ are optimized simultaneously, and $\boldsymbol{q}^*$ is

slightly varied to meet constraints. The difference between $\boldsymbol{q}^*$ and $\hat{\boldsymbol{q}}$ indicates the accuracy of prediction.

In Figure 4.19, the IK solution of target with coordinate $(-40, 40)$ are shown. The posture $\boldsymbol{q}^*$ is obtained through optimization and it satisfies the constraint on end effector. The yellow dashed line is the estimated posture from GPR of forward kinematics based on the optimized feature vector $\boldsymbol{z}$, and the gray region is the 95% confidence interval of posture prediction based on equation 4.20. The algorithm is optimizing the feature vectors under soft constraints that the feature vectors must be close to the high fidelity region as shown in Figure 4.15. Thus the final IK posture here is slightly drifted from the FK prediction to meet the constraints but still in the confidence interval. The red posture is reconstructed from feature vector $\boldsymbol{z}$ by inverse projection with equation (4.24), and it represents the actual posture that corresponds to the feature $\boldsymbol{z}$. The optimized posture is highly similar to the real posture and the later one is also close to the requirement of constraint.

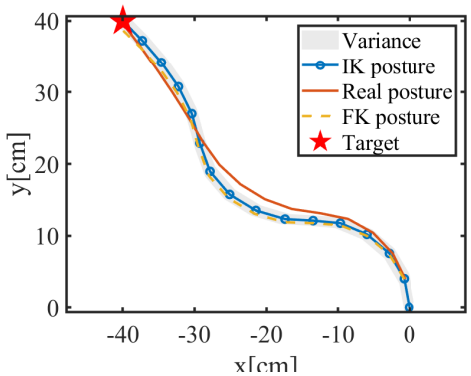
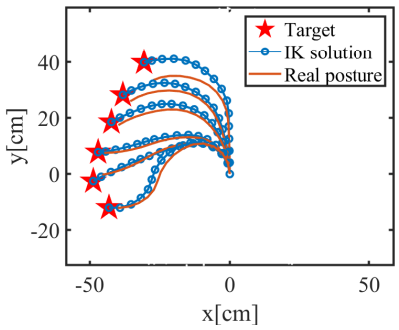
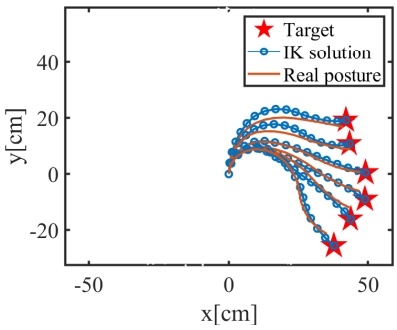


Figure 4.19: The IK solution for target position (-40,40) of end effector. The variance computed from (4.20) is shown in gray strip. The posture from IK is close to the real pose (red) that can be achieved.

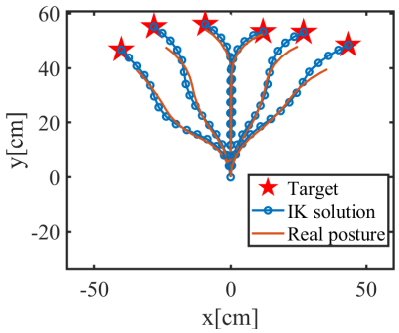
To further test the capability of generalization, we selected a set of targets for end effector in various region of plane, and even out of the workspace soft manipulator shown in Figure 4.20.



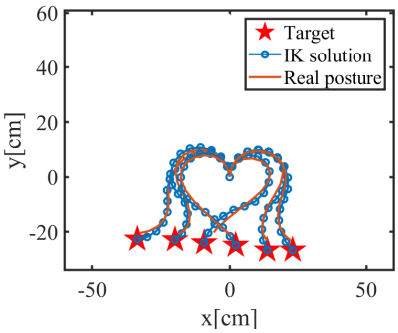
(a) IK solutions for targets located in the left plane.



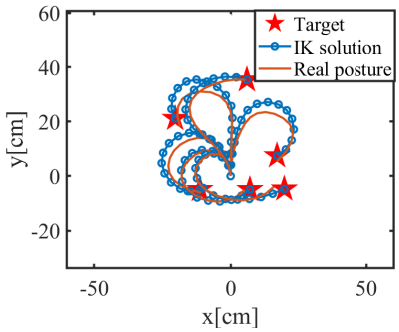
(b) IK solutions for targets located in the right plane.



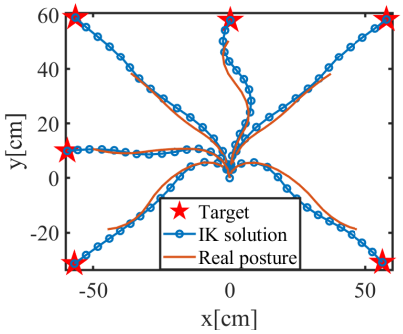
(c) IK solutions for targets located in the top plane.



(d) IK solutions for targets located in the bottom plane.



(e) IK solutions for targets located around origin.



(f) IK solutions for targets located out of workspace.

Figure 4.20: IK solutions for various region in the workspace.

Figure 4.20(a),(b),(c) contains postures with moderate deformation, where the postures are changing continuously with target positions. In Figure 4.20(d),(e), the positions of targets are located in the positions that require more twisted postures while the proposed method still gives IK solution with high accuracy. In last sub-figure, we also tested the case when targets are out of the workspace shown in Figure 4.8, although the solution of IK is intractable the obtained feature vectors still give the closest postures to accomplish the task. Since the body of the soft robot is a curve, and only limited measurement points are available, we cannot give inverse kinematics results with exact length as real robot. Instead, we seek to generate the robot posture of sampling points that is closest to the collected robot postures.

The examples above illustrate the behavior and effectiveness of proposed IK method, while detailed test is still necessary to show the performance. To this end, we tested the IK method on 200 different targets from validation set $\mathcal{D}_V$. The norm of error between IK posture $\boldsymbol{q}^*$ and real posture $\hat{\boldsymbol{q}}$ corresponding to feature vector $\boldsymbol{z}^*$ is computed as shown in Figure 4.21(a). The results here show that the cases with minor error (a)(b)(c) are majority among all tests, and the performance of proposed IK method is stable.

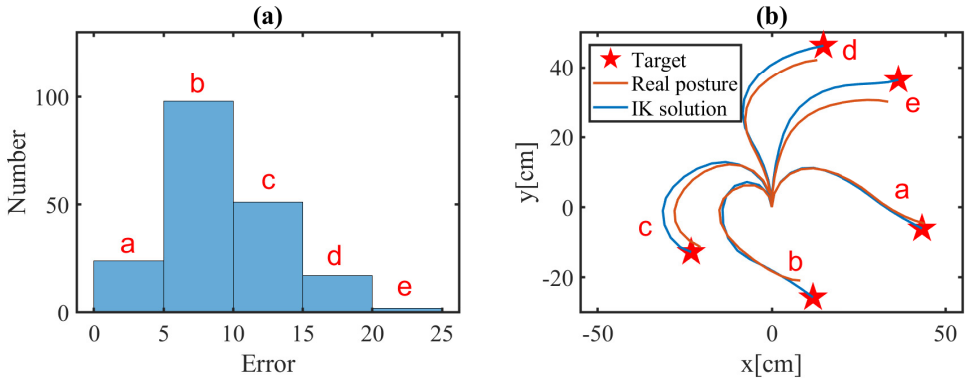


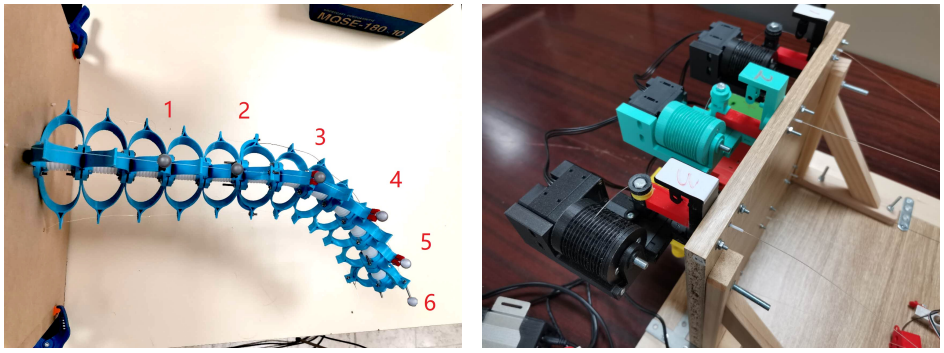
Figure 4.21: The distribution of IK solution error $\|\boldsymbol{q}^* - \hat{\boldsymbol{q}}\|_2$ is illustrated in (a), and 5 solutions corresponding to different error classes are presented in (b).

In Figure 4.21(b), one solution from each error class is shown. For the solution (d)(e) with largest error in the figure, the end-effector of real posture is still close to the target and such error is able to be compensated with feedback controller. While for the solution (a)(b), the result from IK and the actual posture is highly consistent.

4.7 Experimental Validation

4.7.1 Setup

To further validate the effectiveness of proposed method on realistic scenarios, we performed the test on the Echelon robot with high degrees of freedom as shown in Figure 4.22(a). The Echelon is composed by 2 sections with a total length of 52 [cm]. Each section is actuated by Nylon tendons attached at its endpoints. The orientation and position of each section can be changed by pulling the tendons. The vertebrae that form the robot body are 3D printed using regular Polylactic Acid (PLA).

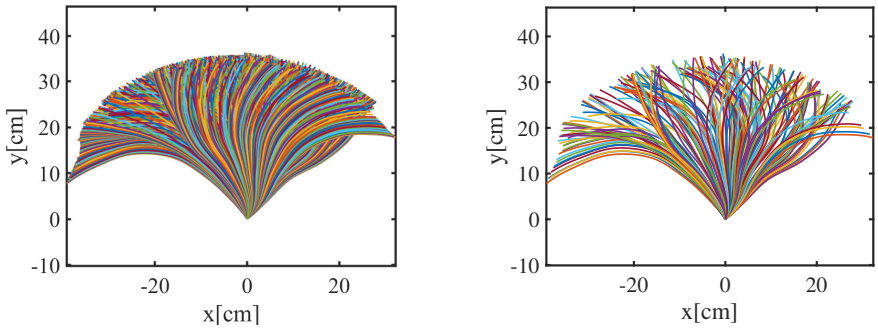


(a) Two-section Echelon robot with 6 optical markers.

(b) 3D printed pulley actuators.

Figure 4.22: Echelon robot and its actuation system.

The robot is actuated by 6 cable actuators with servo motors shown in Figure 4.22(b), and posture data is captured with 6 optical markers to construct posture vector $\boldsymbol{q} \in \mathbb{R}^{12\times 1}$. In total 2000 various postures are collected and the training data set $\mathcal{D}_T = \{\boldsymbol{q}_i, \boldsymbol{z}_i\}, i = 1, \ldots, N$, where the reduced feature vector $\boldsymbol{z} \in \mathbb{R}^{4\times 1}$ is obtained from the POD process as described in section 4.3.1 and the order of feature vector is determined through the histogram of singular values in Figure 4.9. Besides, we also construct a validation data set $\mathcal{D}_V$ with 200 postures out of the training set to test trained IK model as shown in Figure 4.23(b).



(a) Training data set $\mathcal{D}_T$ with 2000 postures.

(b) Validation data set $\mathcal{D}_V$ with 200 postures.

Figure 4.23: Data sets used in the experimental validation.

The performance of IK method with experimental data is studied in the same way as simulation in previous section. The IK method is tested on validation set $\mathcal{D}_V$. The distribution of error between IK posture $\boldsymbol{q}^*$ and real posture $\hat{\boldsymbol{q}}$ is computed as shown in Figure 4.21(a). In the histogram, the term (e) corresponding to larger IK error seems more significant than simulation case, while in fact the maximum error is much lower. Elements from each class are shown in Figure 4.21(b) and even for the worst case (e), the solution of IK is consistent with real postures. The overall accuracy is actually improved compared to the simulation case because the structure is more simple and the maximum deformation is smaller than simulated model. The results here show that the cases with minor error (a)(b)(c) are majority among all tests, and the performance of proposed IK method is stable and accurate.

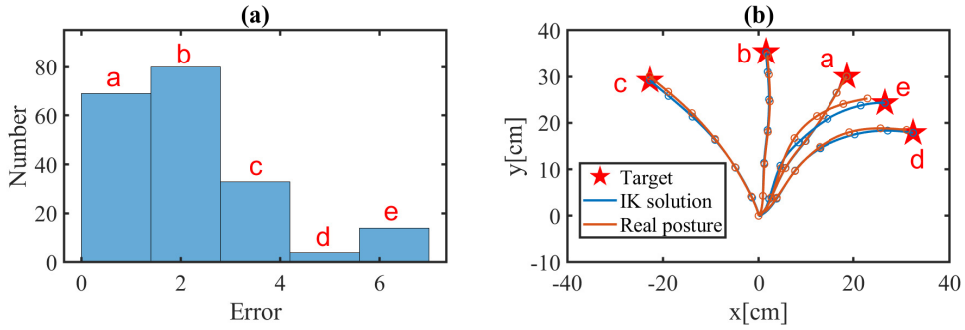


Figure 4.24: The distribution of IK solution error $\|\boldsymbol{q}^* - \hat{\boldsymbol{q}}\|_2$ for experimental data is illustrated in (a), and 5 solutions corresponding to different error classes are presented in (b).

Besides, we also tested the IK method for extreme cases, where the target is out of the workspace as shown in Figure 4.25. The workspace of Echelon robot is roughly shown with the end-effector positions of postures from validation set (blue points in Figure 4.25). It can be observed that, for targets inside the robot workspace (a,b,c), given IK solutions are accurate and consist with real postures. While even for the target outside the workspace, the solutions are still similar to the behavior of soft robot. For target (d), the robot is stretching in the same direction, and for target (e), the robot is also bending toward the target.

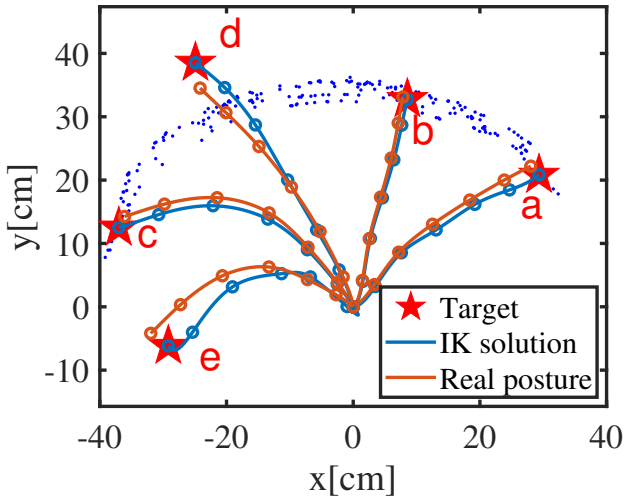


Figure 4.25: The IK solutions for both regular and extreme cases. The blue points are the end-effector positions of postures from validation set.

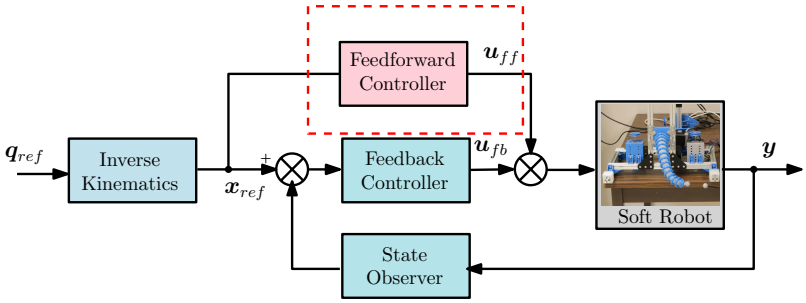
4.8 Summary

In this chapter, we propose a new forward and inverse kinematics framework for soft robots. This allows us to obtain feature vectors from the collected robot postures and use a Gaussian process to learn the hidden structures of the robot postures to realize forward kinematic prediction. To solve the inverse kinematics problem, we transform it into a constrained optimization problem, where the cost function is based on the trained Gaussian process for the similarity measurement between the generated posture and the learned ones. After that, we get solutions that satisfy the constraints and are close to the actual possible postures of the robot. While solving the inverse kinematics, we also get the generalized coordinates corresponding

to the robot's posture as the control target. The proposed method is validated numerically on a three-segments planar soft robot model, and although the robot's model contains redundancy, we can obtain inverse kinematics solutions in each part of the workspace, even including large-deformed twisted postures. Validation based on robot experiment data is performed as well to study the performance on real scenario with more uncertainties and obtained satisfied results. mThe proposed method is able to represent soft robot postures with reduced order generalized coordinates, and the same generalized coordinates are used for the kinematics and dynamics models as for rigid-body robots. However, we cannot directly derive the required feedforward control from this kinematic solution, which will be discussed in the next chapter.

Chapter 5

Perspectives and Conclusion



In the different chapters of this manuscript, we have developed LPV controller and inverse kinematics for soft robot. However, for the control of such nonlinear systems, the trajectory tracking problem is as crucial as the stabilization problem. In order to achieve desirable control performance and ensure tracking stability, it is also essential to introduce a feedforward controller to compensate for the non-linearity behavior of the system. This is the last piece of puzzle in the control diagram above.

However, due to the limitations of the FEM based soft robot model, we must introduce a new method to solve this feedforward problem. In this chapter we proposed a learning-based feedforward control algorithm that can update in the control process. The learning-based control scheme is validated in simulation with a linear system model. Due to the limited time, only preliminary studies have been conducted on this topic.

At the end of the chapter, our contributions and remaining challenges related to dynamic control of soft robots are summarized to concludes the thesis.

5.1 A glimpse of feedforward control

The feedforward control draws attention from both neuroscientists and control researchers. Feedforward control is recognized as a crucial mechanism of the central nervous system which enables coordinates normal movement and how it contributes to motor adaptation and motor learning [28]. For advanced mechatronics systems with critical performance requirement, feedforward is required to push the response to physical limit [116].

The feedforward control is a type of open loop control based on knowledge about the system model and information about the incoming disturbances or desired response. It is calculated directly from desired trajectory instead of tracking error and can be improved with more experience and information. Most high level sports are feedforward competitions, e.g. soccer, gymnastics, archery, where athletes do not have immediate feedback to improve their performance during the competition.

Another example is singing, where a singer is able to control his tone and breathing to sing from a music sheet even without an earphone as feedback. Opera is a good example, typical music sheet is shown in Figure 5.1, the music notes can actually be interpreted as reference output signals from the singer. The process of singing is mostly based on feedforward and we can imagine that a feedback singing would be a disaster because the transient between notes would be either over-damped or under-damped.

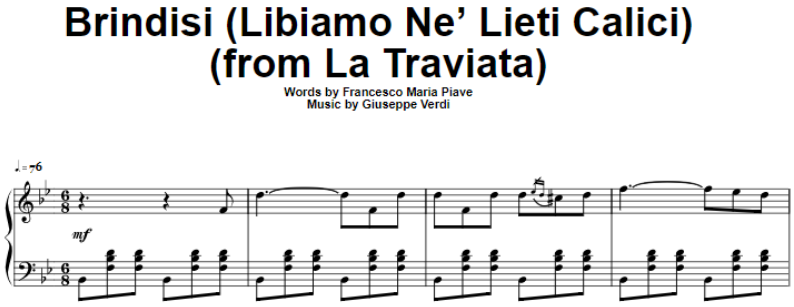


Figure 5.1: This music sheet describes the desired output of the singer.

As discussed in chapter.1, elephants also control their trunk with feedforward scheme, which contributes to the dexterous manipulation and soft response. To give an idea of what is needed in terms of learning, a baby elephant takes more than one year to manage its trunk through countless trials and fails. Developing feedforward scheme for soft robot naturally becomes an interesting track to explore

to improve the control performance, while the realization is complicated for such system without high-quality model. In Chapter 3, we designed a naive feedforward scheme by interpolation inputs of equilibriums ; naive, in the sense that it is based on low model quality and a low speed assumption that may be conservative for more general cases. Although the linearized model has many limitations, we can try to combine it with data to improve the feedforward control. In this chapter, we present directions for representing a function of an unknown feedforward control as well as a model-based method for learning this feedforward function during the control process.

5.1.1 Feedforward as a function

Before illustrating the proposed method, we need to first clarify the nature of feedforward, which can not only be seen as a signal but also as a function. To this end, we refer to the general form of a robotic system:

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{u} \tag{5.1}$$

where $\boldsymbol{q}$ is vector of generalized coordinates, $\mathbf{M}$ is matrix of inertia, $\mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ collects forces and $\boldsymbol{u}$ is control input. Given the desired trajectory $\boldsymbol{q}_d$ and tracking error $\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$, the feed-forward feedback controller (Computed Torque Controller [117]) gives the torque of each input according to the system dynamics with

$$\mathbf{u} = \underbrace{\mathbf{M}\left(\boldsymbol{q}_d\right)\ddot{\boldsymbol{q}}_d + \mathbf{F}\left(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d\right)}_{feedforward} + \underbrace{\mathbf{K}_p\mathbf{e} + \mathbf{K}_v\dot{\mathbf{e}}}_{feedback} \tag{5.2}$$

where the feed-forward part comes from model based information, and feedback gains are designed to fulfill stability conditions [100]. Of course, the feedforward control part is a function of the desired trajectory, and makes use of system model to directly compute required input.

This formulation is actually equivalent to an inverse model using a transfer function, while it is written with differential equations. Unlike rigid robots with precise models, such global non-linear model is currently not accessible for soft robots. The only possible FEM model is of large scale and not accurate. While the feedforward of soft robots can be also treated as a function of general coordinates and its derivatives. The feedforward function can not be obtained analytically but it can be recovered from data.

5.1.2 Related work about feed-forward control

Various methods about feedforward control have been proposed while they have different emphases. In this section, these works are reviewed based on the following aspects:

- Model based or data-driven
- Stability guarantee
- Online learning possibility
- Generalization capability to various trajectories

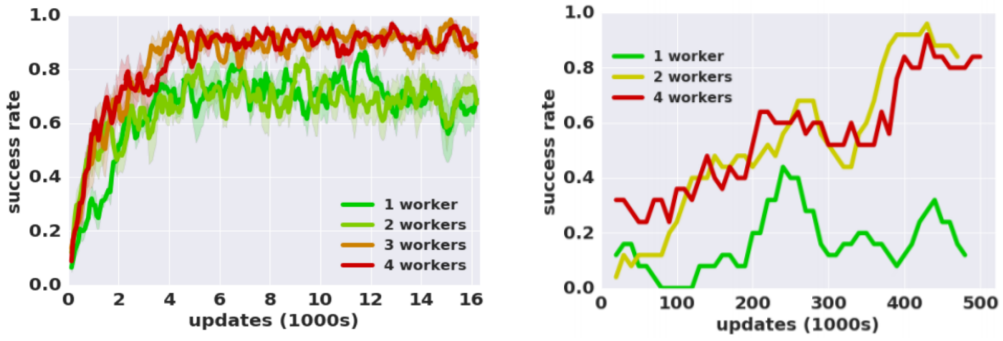
Dictionary-like method

The idea of these methods is to sample control inputs densely and capture the corresponding deformation of the robot. Thereby creating a large dataset of matching shapes and control parameters. Firstly the randomly generated control input $\boldsymbol{u}$, corresponding $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are collected offline to build the regressor $\boldsymbol{u} = g(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ using universal approximators such as neural networks. Then during the control stage, the regressor uses the desired trajectory $\boldsymbol{q}_d$ to deliver the control input. In the work [117], 13622 pairs of input and output data are collected to have comparable performance with PD control for rigid 6-axis manipulator. Furthermore, feedback linearization can also be achieved using such regressor [118] and the regressor can also be used to generate new references from desired trajectory instead of control input [119]. This method does not require system model and can generate to various trajectories, but it must be trained beforehand and its stability cannot be guaranteed. This method takes a direct data-driven approach to learn the kinematics of a robot and bypasses the modeling process, which shows particular interest in soft robot control [52], [120].

Reinforcement learning method

Due to the development of deep learning, reinforcement learning can now be applied to tasks of high complexity. The control input is given by a policy function $\boldsymbol{u} = \mu(\boldsymbol{x}, \boldsymbol{x}_{ref})$ to be learnt according to current system state $\boldsymbol{x}$ and reference $\boldsymbol{x}_{ref}$. The policy function is trained through iterations to maximize the reward given with a reward function $\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{x}_{ref})$. The classical reinforcement learning is designed for the discrete policy game to find the policy that maximizes the reward [121]. The discretization of a continuous problem usually leads to an explosion in the dimension of the search space. A deep neural network can be used as a regressor to obtain

the output of the continuous policy and make reinforcement learning feasible in the context of robot control [122]. A successful RL method consists of an effective algorithm to perform random exploration and policy update together with a well designed tricky reward function. These empirical and trial-based behaviors are also the main limitations for the application of this method. The most influential work about reinforcement learning in robotics in our opinion is the Deep Deterministic Policy Gradient algorithm(DDPG) [122] with real time validation results shown in figure 5.2.



(a) Training in simulation (b) Training with real robots

Figure 5.2: The success rate of RL control in the training process [122]. The number of workers indicates how many models or real robots are trained in parallel. It is worth noting that the convergence of this method is not guaranteed in real time as shown in (b).

Recently some applications of the reinforcement learning technique on the soft robots has been published [123]. The system of one section trunk like robot achieves 98% success rate of reaching after training for 100000 episodes. The reinforcement learning technique is a model-free method and it can be learnt online and generalized to various trajectories. Nevertheless, its effectiveness needs to be improved and no guarantee on the convergence can be obtained formally.

Direct inversion based method

Let us recall the discrete error dynamics with unknown input $\boldsymbol{d}_u$:

$$\boldsymbol{e}_{k+1} = A\boldsymbol{e}_k + B(\boldsymbol{u}_k + \boldsymbol{d}_{u,k}) \tag{5.3}$$

where $\boldsymbol{e}_k = \boldsymbol{x}_{d,k} - \boldsymbol{x}_k$ is the tracking error. Then a direct estimation of $\boldsymbol{d}_u$ can be obtained by

$$\hat{\boldsymbol{d}}_{u,k} = B^{\dagger}(\boldsymbol{e}_{k+1} - A\boldsymbol{e}_k - B\boldsymbol{u}_k) \tag{5.4}$$

where $B^{\dagger}$ is the pseudo-inverse. Generally for a disturbance observer, the unknown input $\boldsymbol{d}_u$ is often formulated as a function of time $\boldsymbol{d}_u(t)$, while it is also possible to consider the feedforward input as a function of the reference trajectory, i.e. $\boldsymbol{d}_u(\boldsymbol{x}_d)$ according to equation 5.2. Since the FF input $\boldsymbol{d}_u(\boldsymbol{x}_d)$ can be estimated point-wise for each point of trajectory $\boldsymbol{x}_d$, it can also be reconstructed by interpolation. Therefore, the interpolated $\hat{\boldsymbol{d}}(\boldsymbol{x})$ can be directly used for compensating in control [124], or taken into account into a MPC controller as in [110]. This method can be applied to various trajectories. It also inspires us about how to formulate the function to be learnt.

Iterative learning control

ILC is used in many motion systems including: additive manufacturing machines, printing systems and wafer stages [116]. It can significantly enhance the performance of equipments that are performing repetitive tasks. Iterative learning control (ILC) is based on the notion that the performance of a system that executes the same task multiple times can be improved by learning from previous iterations. Unlike other control methods, ILC aims at optimizing the control input instead of the controller. The classical formulation of the updating law [93] is:

$$\boldsymbol{u}_i(t) = \boldsymbol{u}_{i-1}(t) + L\boldsymbol{e}_{i-1}(t) \tag{5.5}$$

where i is the number of iteration, $\boldsymbol{u}(t)$ is the control sequence, $\boldsymbol{e}(t)$ is the tracking error, L is the learning gain. The control sequence is updated after each iteration, and the convergence is guaranteed.

Classical ILC is limited to one control sequence and loses the ability to generalize and several approaches have been proposed to improve the generalization properties of ILC like basis functions [125]. Besides, the ILC is only mature for the SISO system, the MIMO case can be dealt with by multiple SISO designs and more general solutions are still under exploration [93] .

5.1.3 Proposed feedforward (FF) Learning Control

The approach proposed in this thesis is inspired by results coming from the field of neuroscience regarding human motor learning mechanisms [28]. Such learning scheme is based on the tracking error and the feedback control from our cerebellum. For a movement requiring precision for the human, the trajectory planning and feedforward

control tasks are performed by the cerebellum, while our vision system is involved in the feedback control to accomplish the movement. The feedforward control signals generated by the cerebellum during our movements inevitably result in tracking errors, and visual feedback helps to compensate errors and complete the specified movement. Meanwhile the cerebellum updates its own feedforward controller based on the feedback error signal [27], which is known as the motor learning mechanism. It combines both feedback and FF controller and the FF controller is updated online based on the feedback one. Such control scheme can be summarized by the following diagram

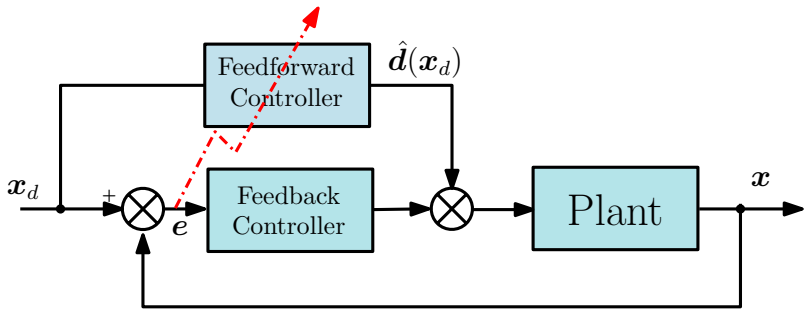


Figure 5.3: The control diagram of feedback error learning. The feedforward controller is updated based on the feedback error. In the ideal case, the feedforward signal is perfect so that no tracking error occurs.

The scheme of human motor learning has following interests for control application:

- Coexistence of feedforward and feedback controller to guarantee performance and robustness
- Feedforward function $\hat{\boldsymbol{d}}(\boldsymbol{x}_d)$ with a possibility to update online and having the capability to generalize to different trajectories(extrapolation)
- Model based method being able to use a priori information of the system
- Reduced feedback gain to reduce stiffness and improve safety

Despite these interesting advantages, there are several challenges to the implementation of this control mechanism. Firstly, the feedforward function cannot be expressed in the form of a parametric function. Secondly, although the feedforward function is considered to be updated based on the feedback error [28], a corresponding update law has to be defined using a mathematical description.

This chapter aims to introduce a new feedforward learning control framework that can achieve online learning of feedforward function while keeping stability guarantee. To this end, we will take profit again of radial basis function network to represent the desired function. The main difficulty associated with this technique is that the updating law of the basis function coefficients should be shown to preserve the control stability.

The main contributions of this work are to represent using radial basis network the feedforward as a function of desired trajectory. Besides, a new parameter updating law is derived from constructed Lyapunov function to update feedforward function. The proposed solution has strong connections with computed torque control [117] and adaptive control [126].

5.1.4 Mathematical formulation of feed-forward learning problem

Let us start from the discrete time linear system:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k.$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n\times 1}, \boldsymbol{u}_k \in \mathbb{R}^{m\times 1}, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$. According to the block diagram 5.1.3, the update process of feed-forward coexists with the feedback control to guarantee the stability. The system is assumed to be stable and here we omit the feedback controller.

The achievable desired trajectory with input $\boldsymbol{u}^*$ is represented with $\boldsymbol{x}_k^*$, then the dynamics for the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}_k^*$ becomes:

$$\boldsymbol{e}_{k+1} = A\boldsymbol{e}_k + B(\boldsymbol{u}_k + \boldsymbol{u}^*) = A\boldsymbol{e}_k + B(\boldsymbol{u}_k + \boldsymbol{d}(\boldsymbol{z}_k)).$$

here $\boldsymbol{z}_k^T = [\boldsymbol{x}_k^{*T}, \boldsymbol{x}_{k+1}^{*T}]$, $\boldsymbol{d}(\boldsymbol{z}_k)$ denotes the unknown input related to the desired trajectory, which will be learnt and compensated during the control process.

With the feed-forward estimation $\boldsymbol{u} = -\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$ as the control input, we have $\Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) = \boldsymbol{d}(\boldsymbol{z}_k) - \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$, and the close loop system becomes:

$$\boldsymbol{e}_{k+1} = A\boldsymbol{e}_k + B\Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k). \tag{5.6}$$

The index k of estimation $\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$ means that it is updating with the time and the argument $\boldsymbol{z}_k$ is related to the desired trajectory. Since the $\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$ can not be represented as a parametric function, we will discuss its possible nature and introduce proper method in the next section.

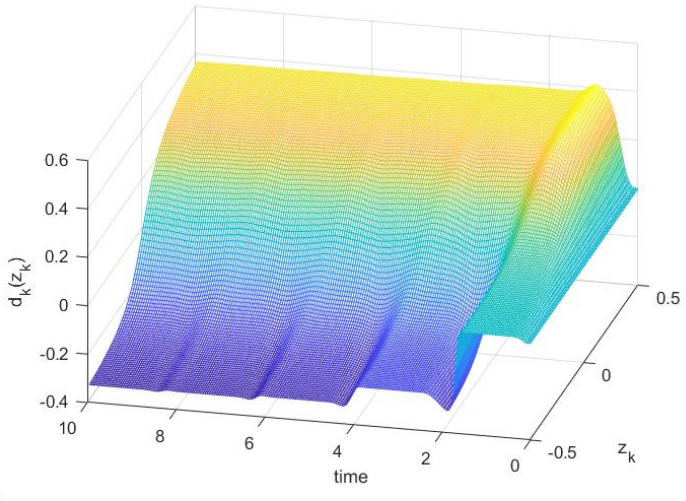


Figure 5.4: The estimation process of function $\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$ on the first dimension of $\boldsymbol{z}_k$ from examples in section 5.1.9. The shape of the function is updating with time.

5.1.5 Function representation

Since we do not have the exact information about the function that describes the input-output relationship, the most suitable way is to approximate it using universal regressors like neural network or Gaussian process [113].

In the related work [127] and [126], integral transformation and Gaussian network are used to express the function of adaptive control law. Among all these non-parametric methods, the techniques based on radial basis functions seem to us again appropriate. Obviously for their universal approximation property [128] but also because the functions are differentiable which is necessary to define the adaptation law

The evaluation of RBF network is given by:

$$Y(\boldsymbol{x}) = \sum_{k=1}^{N} K(\boldsymbol{x}, \boldsymbol{x}_k)\boldsymbol{c}_k \tag{5.7}$$

where $Y(\boldsymbol{x})$ is the prediction value of input $\boldsymbol{x}$, $\boldsymbol{c}_k$ are constant parameters, $K(\boldsymbol{x}, \boldsymbol{x}_k)$ is the Radial basis function kernel (Gaussian kernel) as formulated below:

$$K(\boldsymbol{x}, \boldsymbol{x}_k) = e^{-r/2\sigma^2}, r = (\boldsymbol{x} - \boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k),$$

and $\boldsymbol{x}_k$ is the center of k-th Gaussian kernel. Here in Figure 5.5 we recall the illustration used chapter 3 to show the mechanism of such kernel based method.

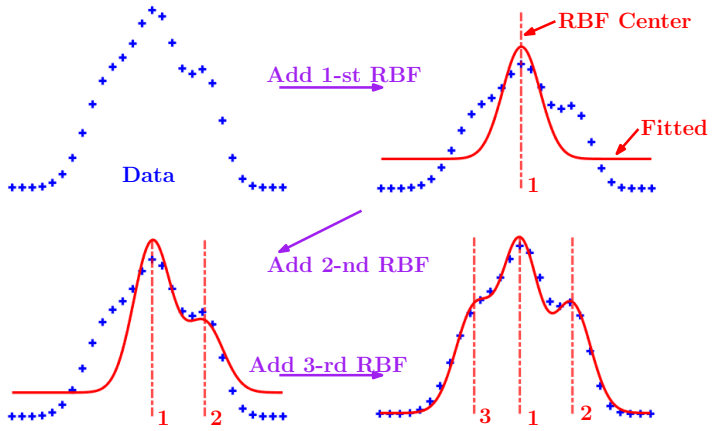


Figure 5.5: Representation of function with 3 RBFs. The predicted value of the function is given by the weight sum of the known data points, and the weight is related to the distance,

The Gaussian function is a typical membership function used for approximation techniques (fuzzy systems and/or neural networks) and it is also a positive definite kernel function of a Hilbert's space $\mathcal{H}_k$ [129]. The result from kernel method indicates that any continuous function $f(\boldsymbol{x}) \in \mathcal{H}_k$ can be expressed as the combination of a finite number of functions:

$$f(\boldsymbol{x}) = \sum_{k=1}^{N} K(\boldsymbol{x}, \boldsymbol{x}_k)\boldsymbol{c}_k. \tag{5.8}$$

Therefore it exists a Gaussian kernel such that function $\boldsymbol{d}(\boldsymbol{z}_k)$ and its estimation $\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$ can be approximated as:

$$\begin{aligned} \boldsymbol{d}(\boldsymbol{z}_k) &= \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)\boldsymbol{c}(\boldsymbol{z}_i), \\ \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) &= \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i), \\ \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) = \boldsymbol{d}(\boldsymbol{z}_k) - \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) &= \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)\Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i). \end{aligned} \tag{5.9}$$

where Ω is the set of desired trajectory $\boldsymbol{z}_k$, $K(.,.)$ is the Gaussian kernel, and $\boldsymbol{c}(\boldsymbol{z}_i)$ and $\boldsymbol{c}_k(\boldsymbol{z}_i)$ are the coefficients and $\Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) = \boldsymbol{c}(\boldsymbol{z}_i) - \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)$ is the error between coefficients. This method is defined on the set of desired trajectories Ω but not a certain trajectory. The feedforward can be trained for various trajectories and extrapolated from trained ones. Besides, with the normalized RBF network technique and several similar methods [113], [130], [131],we can also have the convex representation with

$$\sum_{\boldsymbol{z}_i \in \Omega} K(z, \boldsymbol{z}_i) = 1.$$

5.1.6 Updating law

To realize the learning of FF function, a update scheme of $\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$ is required based on the RBF representation. Inspired from [127], and considering the special form of $\boldsymbol{d}(\boldsymbol{z}_k)$ in (5.8), the update is performed on the coefficients $\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)$ as:

$$\hat{\boldsymbol{c}}_{k+1}(\boldsymbol{z}_i) = \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) + K(\boldsymbol{z}_i, \boldsymbol{z}_k) L \boldsymbol{e}_{k+1},$$

here L is the learning gain matrix to be designed $K(\boldsymbol{z}_i, \boldsymbol{z}_k)$ is the value of the RBF function, which indicates its corresponding contribution. Therefore, the corresponding updating law of coefficient error $\Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)$ becomes:

$$\Delta\hat{\boldsymbol{c}}_{k+1}(\boldsymbol{z}_i) = \Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) - K(\boldsymbol{z}_i, \boldsymbol{z}_k) L \boldsymbol{e}_{k+1}. \tag{5.10}$$

At each step, the parameters of the RBF network are updated according to their contribution to the tracking error. The learning control scheme is summarized in algorithm.3 and the asymptotic stability of both tracking control and feed-forward learning are discussed in next section.

5.1.7 Stabilization

To stabilize the system (5.6) with learning update law (5.10), we can derive the stability conditions using Lyapunov theory [50]. To this end, we propose a Lyapunov function with two parts:

$$V = V_1 + V_2.$$

where

$$V_1 = \boldsymbol{e}_k^T P \boldsymbol{e}_k. \tag{5.11}$$

describes the error for tracking control with a positive definite matrix $P = P^T$. The second part of the Lyapunov function V_2 is related to the estimation error of the

Algorithm 3 Online estimation of feed-forward

1: Design a linear feedback controller to have A matrix stable
2: $k \leftarrow 1$
3: Initialize coefficients $\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)$ of RBF network in (5.9) with zeros
4: **repeat**
5: Evaluate $\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) = \sum_{\boldsymbol{z}_i \in Z} K(\boldsymbol{z}_i, \boldsymbol{z}_k)\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)$ as feed-forward input
6: Send to the plant and wait ΔT for the next time step
7: Obtain the measurement for the tracking error $\boldsymbol{e}_{k+1}$
8: **for** $\boldsymbol{z}_i \in \Omega$ **do**
9: $\hat{\boldsymbol{c}}_{k+1}(\boldsymbol{z}_i) = \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) + K(\boldsymbol{z}_i, \boldsymbol{z}_k)L\boldsymbol{e}_{k+1}$ {Update all the coefficients}
10: **end for**
11: $k \leftarrow k + 1$
12: **until** Desired performance is met

feedforward contribution (5.8). Consider the function $V_{\Delta d}$:

$$V_{\Delta d} = \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T Q \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$$

with $Q > 0$. Inspired from [126], it is possible to make the connection between the function represented on the RBF network and on its coefficients. After substituting (5.9) into $V_{\Delta d}$ and using Cauchy–Schwartz inequality, we have

$$\begin{aligned} V_{\Delta d} &= \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T Q \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \\ &= \left(\sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)\Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)\right)^T Q \left(\sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)\Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)\right) \\ &\leq \left(\sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)^2\right)\left(\sum_{\boldsymbol{z}_i \in \Omega} \Delta\hat{\boldsymbol{c}}_k^T(\boldsymbol{z}_i) Q \Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)\right) \\ &\leq \sum_{\boldsymbol{z}_i \in \Omega} \Delta\hat{\boldsymbol{c}}_k^T(\boldsymbol{z}_i) Q \Delta\hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) \coloneqq V_2, \end{aligned} \tag{5.12}$$

A larger Lyapunov function $V_2 > V_{\Delta d}$ is obtained. It is similar to the Parseval's theorem that the sum (or integral) of the square of a function is equal to the sum (or integral) of the square of its Fourier transform.

The difference of V_1 can be obtained from (5.11)

$$\begin{aligned} \Delta V_1 &= \boldsymbol{e}_{k+1}^T P \boldsymbol{e}_{k+1} - \boldsymbol{e}_k^T P \boldsymbol{e}_k \\ &= \boldsymbol{e}_k^T (A^T P A - P)\boldsymbol{e}_k + 2\boldsymbol{e}_k^T A^T P B \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T P B \Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k). \end{aligned} \tag{5.13}$$

The difference of V_2 in (5.12) is

$$\begin{aligned}
\Delta V_2 &= \sum_{\boldsymbol{z}_i \in \Omega} [\Delta \hat{\boldsymbol{c}}_{k+1}^T(\boldsymbol{z}_i) Q \Delta \hat{\boldsymbol{c}}_{k+1}(\boldsymbol{z}_i) - \Delta \hat{\boldsymbol{c}}_k^T(\boldsymbol{z}_i) Q \Delta \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)] \\
&= \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)[-2\boldsymbol{e}_{k+1}^T L^T Q \Delta \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i)] + \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)^2 \boldsymbol{e}_{k+1}^T L^T Q L \boldsymbol{e}_{k+1} \\
&= -2\boldsymbol{e}_{k+1}^T L^T Q \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k) \Delta \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) + \boldsymbol{e}_{k+1}^T L^T Q L \boldsymbol{e}_{k+1} \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k)^2 \\
&\leq -2\boldsymbol{e}_{k+1}^T L^T Q \sum_{\boldsymbol{z}_i \in \Omega} K(\boldsymbol{z}_i, \boldsymbol{z}_k) \Delta \hat{\boldsymbol{c}}_k(\boldsymbol{z}_i) + \boldsymbol{e}_{k+1}^T L^T Q L \boldsymbol{e}_{k+1} (\textit{with convexity}) \\
&= -2\boldsymbol{e}_{k+1}^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + \boldsymbol{e}_{k+1}^T L^T Q L \boldsymbol{e}_{k+1} \\
&= -2\boldsymbol{e}_k^T A^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) - 2\Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \\
&\quad + \boldsymbol{e}_k^T A^T L^T Q L A \boldsymbol{e}_k + 2\boldsymbol{e}_k^T A^T L^T Q L B \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + \Delta \hat{\boldsymbol{d}}_k^T(\boldsymbol{z}_k) B^T L^T Q L B \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \\
&= \boldsymbol{e}_k^T A^T L^T Q L A \boldsymbol{e}_k + 2\boldsymbol{e}_k^T \left(A^T L^T Q L B - A^T L^T Q \right) \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \\
&\quad + \Delta \hat{\boldsymbol{d}}_k^T(\boldsymbol{z}_k) [B^T L^T Q L B - (B^T L^T Q + Q L B)] \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)
\end{aligned}$$

we can rewrite in matrix form

$$\Delta V_2 \leq \begin{bmatrix} \boldsymbol{e}_k \\ \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \end{bmatrix}^T \begin{bmatrix} A^T L^T Q L A & A^T L^T Q L B - A^T L^T Q \\ * & B^T L^T Q L B - (B^T L^T Q + Q L B) \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_k \\ \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \end{bmatrix} = \Delta V_2^* \tag{5.14}$$

Now we can combine ΔV_2^* and ΔV_1 in (5.13) to have

$$\Delta V \leq \begin{bmatrix} A^T L^T Q L A + A^T P A - P & A^T L^T Q L B + A^T P B - A^T L^T Q \\ * & B^T L^T Q L B + B^T P B - (B^T L^T Q + Q L B) \end{bmatrix} < 0.$$

It can be reformulated as

$$\begin{bmatrix} A^T P A - P & A^T P B - A^T L^T Q \\ (*) & B^T P B - \left(B^T L^T Q + Q L B \right) \end{bmatrix} + \begin{bmatrix} A^T L^T Q \\ B^T L^T Q \end{bmatrix} Q^{-1} \begin{bmatrix} Q L A & Q L B \end{bmatrix} < 0.$$

Using the Schur complement, we can obtain

$$\begin{bmatrix} A^T P A - P & A^T P B - A^T L^T Q & A^T L^T Q \\ * & B^T P B - (B^T L^T Q + Q L B) & B^T L^T Q \\ * & * & -Q \end{bmatrix} < 0.$$

Using the change of variable $M = QL$

$$\begin{bmatrix} A^TPA - P & A^TPB - A^TM^T & A^TM^T \\ * & B^TPB - (B^TM^T + MB) & B^TM^T \\ * & * & -Q \end{bmatrix} < 0. \tag{5.15}$$

The feedforward learning control scheme is stable if LMI (5.15) with respect to the decision variables P, M, Q with the conditions $P = P^T > 0$ and $Q = Q^T > 0$ are feasible. Here the first diagonal part $A^TPA - P$ is negative due to the pre-designed feedback control law.

5.1.8 Disturbance Rejection

In the principle of proposed feed-forward learning algorithm, all tracking errors are attributed to the error of the feed-forward function. However, in practice, there are disturbances or uncertainty outside the control channel, so we need to design a feed-forward learning law that can reject mismatch disturbances. The system with mismatch disturbances can be formulated as

$$\boldsymbol{e}_{k+1} = A\boldsymbol{e}_k + B\Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + D\boldsymbol{w}_k. \tag{5.16}$$

where $D \in \mathbb{R}^{n\times p}, \boldsymbol{w}_k \in \mathbb{R}^{p\times 1}$. The goal is to find the best $L_2 \to L_2$ gain to minimize the impact of disturbance $\boldsymbol{w}_k$ on feed-forward estimation $\Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)$, i.e.

$$\sup_{\boldsymbol{w}_k \neq 0} \frac{||\Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)||_2}{||\boldsymbol{w}_k||_2} \leq \gamma.$$

Theorem 4. The discrete system (5.16) with feed-forward learning law (5.10) are globally stable and the attenuation of the disturbance $\boldsymbol{w}$ to estimation error is at least γ, if there exist matrices $P > 0, Q > 0$ and M such that the LMI

$$\begin{bmatrix} A^TPA-P & A^TPB-A^TM^T & A^TPD & A^TM^T \\ (*) & I+B^TPB-\left(B^TM^T+MB\right) & B^TPD-MD & B^TM^T \\ (*) & (*) & -\gamma^2 I+D^TPD & D^TM^T \\ (*) & (*) & (*) & -Q \end{bmatrix} < 0$$

holds.

Proof: Consider the previous Lyapunov function with performance criteria such that

$$\Delta V_k + \Delta\hat{\boldsymbol{d}}_k^T(\boldsymbol{z}_k)\Delta\hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) - \gamma^2\boldsymbol{w}_k^T\boldsymbol{w}_k < 0$$

After summing this expression from $k = 0$ to $k = \tau$ we have

$$V_\tau - V_0 < \sum_{k=0}^{\tau} (\gamma^2 \boldsymbol{w}_k^T \boldsymbol{w}_k - \Delta \hat{\boldsymbol{d}}_k^T(\boldsymbol{z}_k) \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k))$$

With initial condition $V_0 = 0$, we have

$$\sum_{k=0}^{\tau} \Delta \hat{\boldsymbol{d}}_k^T(\boldsymbol{z}_k) \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + V_\tau < \sum_{k=0}^{\tau} \gamma^2 \boldsymbol{w}_k^T \boldsymbol{w}_k$$

Taking into account that $V_\tau > 0$, we can obtain

$$||\Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)||_2 < \gamma ||\boldsymbol{w}_k||_2.$$

To obtain the stability condition, we continue with previous Lyapunov functions with new system The difference of V_1 in (5.11) with system (5.16) is

$$\begin{aligned}
\Delta V_1 &= \boldsymbol{e}_{k+1}^T P \boldsymbol{e}_{k+1} - \boldsymbol{e}_k^T P \boldsymbol{e}_k \\
&= \boldsymbol{e}_k^T (A^T P A - P) \boldsymbol{e}_k + 2 \boldsymbol{e}_k^T A^T P B \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + 2 \boldsymbol{e}_k^T A^T P D \boldsymbol{w}_k \\
&\quad + 2 \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T P D \boldsymbol{w}_k + \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T P B \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + \boldsymbol{w}_k^T D^T P D \boldsymbol{w}_k \quad (5.17)
\end{aligned}$$

The difference of V_2 in (5.12) with system (5.16) is

$$\begin{aligned}
\Delta V_2 &\leq -2 \boldsymbol{e}_{k+1}^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + \boldsymbol{e}_{k+1}^T L^T Q L \boldsymbol{e}_{k+1} \\
&= -2 \boldsymbol{e}_k^T A^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) - 2 \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) - 2 \boldsymbol{w}_k^T D^T L^T Q \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \\
&\quad + \boldsymbol{e}_k^T A^T L^T Q L A \boldsymbol{e}_k + 2 \boldsymbol{e}_k^T A^T L^T Q L B \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + 2 \boldsymbol{e}_k^T A^T L^T Q L D \boldsymbol{w}_k \\
&\quad + 2 \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T L^T Q L D \boldsymbol{w}_k + \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T B^T L^T Q L B \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + \boldsymbol{w}_k^T D^T L^T Q L D \boldsymbol{w}_k \\
&= \boldsymbol{e}_k^T A^T L^T Q L A \boldsymbol{e}_k + 2 \boldsymbol{e}_k^T (A^T L^T Q L B - A^T L^T Q) \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) + 2 \boldsymbol{e}_k^T A^T L^T Q L D \boldsymbol{w}_k \\
&\quad + 2 \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T (B^T L^T Q L D - Q L D) \boldsymbol{w}_k \\
&\quad + \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k)^T (B^T L^T Q L B - (B^T L^T Q + Q L B)) \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) \\
&\quad + \boldsymbol{w}_k^T D^T L^T Q L D \boldsymbol{w}_k.
\end{aligned}$$

Combining $\Delta V_1, \Delta V_2$, we have $\Delta V_2 + \Delta V_1 + \Delta \hat{\boldsymbol{d}}_k^T(\boldsymbol{z}_k) \Delta \hat{\boldsymbol{d}}_k(\boldsymbol{z}_k) - \gamma^2 \boldsymbol{w}_k^T \boldsymbol{w}_k < 0$, which is

$$\begin{bmatrix}
A^T L^T Q L A + A^T P A - P & A^T L^T Q L B - A^T L^T Q + A^T P B & A^T P D + A^T L^T Q L D \\
* & I + B^T P B + B^T L^T Q L B - (B^T L^T Q + Q L B) & B^T P D + B^T L^T Q L D - Q L D \\
* & * & -\gamma^2 I + D^T P D + D^T L^T Q L D
\end{bmatrix} < 0.$$

It can be reformulated as

$$\begin{bmatrix} A^TPA-P & A^TPB-A^TL^TQ & A^TPD \\ (*) & I+B^TPB-\left(B^TL^TQ+QLB\right) & B^TPD-QLD \\ (*) & (^*) & -\gamma^2 I+D^TPD \end{bmatrix} + \begin{bmatrix} A^TL^TQ \\ B^TL^TQ \\ D^TL^TQ \end{bmatrix} Q^{-1} \begin{bmatrix} QLA & QLB & QLD \end{bmatrix} < 0.$$

Using the Schur complement and the change of variable $M = QL$, we have

$$\begin{bmatrix} A^TPA-P & A^TPB-A^TM^T & A^TPD & A^TM^T \\ (*) & I+B^TPB-\left(B^TM^T+MB\right) & B^TPD-MD & B^TM^T \\ (*) & (*) & -\gamma^2 I+D^TPD & D^TM^T \\ (*) & (*) & (*) & -Q \end{bmatrix} < 0$$

which is the condition for the theorem.

5.1.9 Numerical Example and Summary

Consider a linear dynamic system:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k.$$

where

$$A = \begin{bmatrix} 0.76 & 0.06 \\ 0.06 & 0.89 \end{bmatrix}, B = \begin{bmatrix} -0.006 \\ 0.033 \end{bmatrix}.$$

The desired trajectory $\boldsymbol{x}_d$ is generated with $\boldsymbol{u}_k = \sin(3k)$. The feedback controller corresponds to the linear feedback $\boldsymbol{u} = K\boldsymbol{x}$ with $K = [1.201\ 26.188]$. The performance of the feedback controller is shown in Figure 5.6.

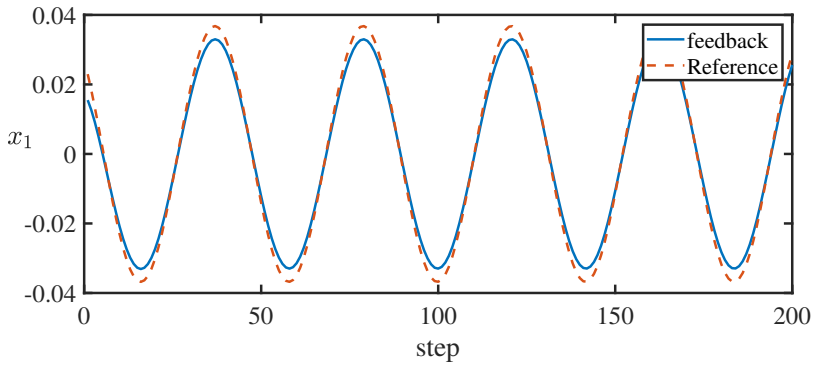


Figure 5.6: Trajectory tracking with the feedback controller.

As expected, Figure 5.6 shows that the system can keep tracking the desired trajectory. However, since there is only a linear feedback control, there is a tracking error repeated during each iteration.

Using the proposed method, a feedforward learning gain is obtained from LMI (5.15) as $L = [-0.29\ 10.39]$. Applying feedforward learning control scheme described in Algorithm 3 together with the feedback controller, the tracking performance is shown in Figure 5.7.

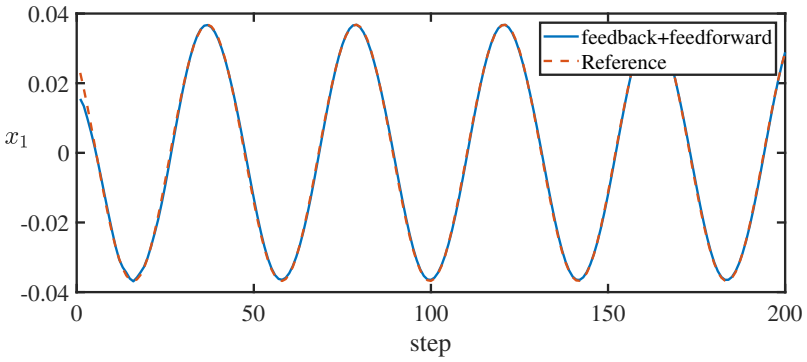


Figure 5.7: Trajectory tracking with proposed feedforward learning scheme.

It can be observed from the figure that, the tracking error decreases rapidly with trajectory repetition. As proven in section 5.1.7, The feedforward learning process is asymptotically stable.

Based on the basis function representation of feedforward function, we developed an updating law based on the coefficients of the RBF network together with Lyapunov stability theory. In our point of view, such feed-forward can be seen as an extension of unknown input disturbance observer. The unknown input is treated as a function of states instead of time. This method can update the feed-forward of the system according to a simplified model (linear or LPV for example) model, which is ideal for nonlinear system like soft robot. Effectively, a global nonlinear model of soft robot is not accessible, while linear models can be obtained from FEM model.

5.2 Future Works

Soft robot as a control object contains rich problems and we were only able to discuss a small part of them in the thesis. Due to the complexity of such system,

the methods proposed in the thesis are not completely mature and worth further exploration.

The first focus is LPV modeling of soft robots. The proposed method uses the set of collected models to build an interpolated model, and gradually increases the number of models according to the modeling error. Although this modeling methodology can represent a large number of systems with fewer models, it is an heuristic method that may introduced conservatism into the obtained model. For instance, the process of adding iteratively new models may lead to redundancy in the models. It is worth studying this modeling methodology further on; for example, adding pruning steps to achieve a balance between model accuracy and complexity.

For the LPV controller synthesis, taking profit both of the discrete system representation and the preservation of the system structure, more choices are available for its design to enlarge the set of suitable and performant solutions. For example, the use of powerful non-quadratic Lyapunov function to design the controller can easily come at hand, keeping in mind that there will be a trade-off between complexity of the conditions and complexity of the Lyapunov function [132]. Less conservative Lyapunov function can also be constructed in a polytopic way to have weaker assumptions on varying parameters [133].

In addition, for the scheduling variables of the LPV model, we use the measurable end-effector positions. According to the nature of soft robot, we can conclude that the model variation is actually related to its shape, and the different shapes of the robot make its input and output behavior different. Therefore, subsequent research can study how to find variables that better represent the shape of the robot. They can be calculated from known states or we can adopt latest measurement methods such as embedded optical fiber as bending sensors [134], etc. In the case of having uncertainties in the estimation of such variables, the closed loop stability can still be guaranteed using a separate design of the observer and controller [135].

For the feed-forward algorithm, we developed the basic framework of controller design, while experiments are only carried out on linear simulation models. It will be necessary to validate on real robotic systems in the future. The linear framework of feed-forward can also be generalized to nonlinear with LPV method to achieve feed-forward learning of robots in the entire workspace. Our long term target is to combine all the proposed algorithms as a complete robot control framework to achieve dexterous soft robot manipulation.

5.3 General Conclusion

Currently, the control of soft robot is still under low speed and far from typical dexterous movement shown in Figure 5.8. While the FEM based modeling, model reduction, and linear control methods of soft robots proposed by the previous works of the team have brought up a new possibility for solving this difficult problem. Based on these previous works, this thesis makes several explorations towards dynamic and task-oriented control of soft robots including modeling, non-linear and feed-forward control as well as inverse kinematics.

(a) The robotic arm catching a ball thrown to it [136].

(b) Robot and human collaborate to assemble a desktop [137].

Figure 5.8: Two typical dexterous movements of robotic system.

The main contributions of this thesis can be summarized in the following categories:

Dynamic modeling To obtain a control oriented model with larger feasible region and higher quality, we propose a new projector for the proper orthogonal decomposition (POD) algorithm to generate a set of linearized reduced-order models representing the local behavior of the soft robot at different operating points within the workspace. With a unified POD projector, not only can the order of these linearized models be significantly reduced, but also their mechanical structure and stability properties can be preserved for LPV modeling and control design. Next, using radial basis function (RBF) networks, we propose an iterative training method to build a LPV model of the robot by interpolating a selected subset of linearized models. The iterative training method can reduce the correlation between linear models, thereby utilizing fewer models to approximate the nonlinear model of the robot with specified interpolation error.

Dynamic control Based on the proposed modeling method, we introduced the equivalent-input-disturbance-based scheme for dynamic tracking control, which is composed of three core components: feedforward control, disturbance-estimator control and feedback control. The feedforward control is designed to account for the effects of the trajectory reference and the time-varying affine term, issued from the FEM-based model linearization. The disturbance-estimator control is obtained from a generalized proportional integral LPV observer, which also provides the estimated reduced-order states for feedback control. The observer-based feedback control design is reformulated as a convex optimization problem under linear matrix inequality (LMI) constraints. The globally uniformly ℓ_∞ stability of the closed-loop LPV robot model is guaranteed by means of Lyapunov stability theory. The effectiveness of the proposed dynamic control framework is firstly demonstrated in the linear case via both high-fidelity SOFA simulations and experimental validations, performed on two soft robots with different natures. Afterwards, detailed experiment tests are conducted under several scenarios with small and large deformations of soft robot to show the effectiveness of the proposed LPV tracking control framework. Comparative studies with linear EID-based controller, PID controller and an iterative learning controller are also conducted to emphasize the interests of the proposed nonlinear control method.

Inverse kinematics To solve the problem of motion planning for soft robots, we propose a hybrid approach based on both model and data. Starting from the model-reduction approach, a reduced representation of robot postures based on feature vectors is developed and we reveal its connection with modal decomposition. The feature vectors for posture representation are obtained from the model, while a Gaussian process is introduced to establish a mapping between measured robot configurations and feature vectors. This step is also known as forward kinematics. The inverse kinematics of the robot is then formulated as an optimization problem with constraints, where the constraints are derived from the robot task. This optimization algorithm aims to generate the posture that is most similar to the measured ones while satisfying the constraints. A large amount of posture data of the robot in both simulation and experiment is collected, and verification of both forward and inverse kinematics shows satisfying performance.

In addition, a preliminary exploration of the feed-forward control that can achieve better dynamic control performance is presented. We also developed the soft robot platforms and digital models used in the thesis are with our partners, including design and manufacture, actuator and sensor setup, and drivers on computer-side.

Bibliography

[1] D. Rus and M. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[2] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, Jan. 1, 2008.

[3] M. Wehner, R. L. Truby, D. J. Fitzgerald, *et al.*, "An integrated design and fabrication strategy for entirely soft, autonomous robots," *Nature*, vol. 536, no. 7617, pp. 451–455, 7617 Aug. 2016.

[4] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics*, vol. 1, no. 1, eaah3690, Dec. 6, 2016.

[5] T. Umedachi, V. Vikas, and B. A. Trimmer, "Softworms: The design and control of non-pneumatic, 3D-printed, deformable robots," *Bioinspir. Biomim.*, vol. 11, no. 2, p. 025 001, Mar. 2016.

[6] J. Cao, W. Liang, Y. Wang, H. Lee, J. Zhu, and Q. Ren, "Control of a soft inchworm robot with environment adaptation," *IEEE Trans. Indus. Electron.*, vol. 67, no. 5, pp. 3809–3818, 2020.

[7] B. Gamus, A. D. Gat, and Y. Or, "Dynamic Inchworm Crawling: Performance Analysis and Optimization of a Three-Link Robot," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 111–118, Jan. 2021.

[8] C. Marchese, "Biodiversity hotspots: A shortcut for a more complicated concept," *Global Ecology and Conservation*, vol. 3, pp. 297–309, Jan. 1, 2015.

[9] J. Shintake, V. Cacucciolo, D. Floreano, and H. Shea, "Soft Robotic Grippers," *Advanced Materials*, vol. 30, no. 29, p. 1 707 035, 2018.

[10] A. Jusufi, D. M. Vogt, R. J. Wood, and G. V. Lauder, "Undulatory Swimming Performance and Body Stiffness Modulation in a Soft Robotic Fish-Inspired Physical Model," *Soft Robotics*, vol. 4, no. 3, pp. 202–210, Sep. 2017.

[11] M. Rolf and J. J. Steil, "Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1147–1160, Jun. 2014.

[12] S. Li, A. Kruszewski, T.-M. Guerra, and A.-T. Nguyen, "Equivalent-input-disturbance-based dynamic tracking control for soft robots via reduced-order finite-element models," *IEEE/ASME Trans. Mechatron.*, pp. 1–12, 2022.

[13] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," vol. 26, no. 7, pp. 709–727, 2012.

[14] Z. Xie, A. G. Domel, N. An, *et al.*, "Octopus Arm-Inspired Tapered Soft Actuators with Suckers for Improved Grasping," *Soft Robotics*, vol. 7, no. 5, pp. 639–648, Oct. 2020.

[15] T. Morales Bieze, A. Kruszewski, B. Carrez, and C. Duriez, "Design, implementation, and control of a deformable manipulator robot based on a compliant spine," *The International Journal of Robotics Research*, vol. 39, no. 14, pp. 1604–1619, Dec. 1, 2020.

[16] B. Jones and I. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, 2006.

[17] Z. J. Patterson, A. P. Sabelhaus, and C. Majidi, "Robust Control of a Multi-Axis Shape Memory Alloy-Driven Soft Manipulator," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2210–2217, Apr. 2022.

[18] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-Driven Control of Soft Robots Using Koopman Operator Theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, Jun. 2021.

[19] G. Fang, X. Wang, K. Wang, *et al.*, "Vision-Based Online Learning Kinematic Control for Soft Robots Using Local Gaussian Process Regression," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1194–1201, Apr. 2019.

[20] J. Shoshani, "Proboscidea (Elephants)," in *eLS*, American Cancer Society, 2001.

[21] P. Dagenais, S. Hensman, V. Haechler, and M. C. Milinkovitch, "Elephants evolved strategies reducing the biomechanical complexity of their trunk," *Current Biology*, vol. 31, no. 21, 4727–4737.e4, Nov. 8, 2021.

[22] W. M. KIER and K. K. SMITH, "Tongues, tentacles and trunks: The biomechanics of movement in muscular-hydrostats," *Zoological Journal of the Linnean Society*, vol. 83, no. 4, pp. 307–324, Apr. 1, 1985.

[23] I. Walker and M. Hannan, "A novel 'elephant's trunk' robot," in *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (Cat. No.99TH8399)*, Sep. 1999, pp. 410–415.

[24] J. Yang, E. P. Pitarch, J. Potratz, S. Beck, and K. Abdel-Malek, "Synthesis and analysis of a flexible elephant trunk robot," *Advanced Robotics*, vol. 20, no. 6, pp. 631–659, Jan. 1, 2006.

[25] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamics for variable length multisection continuum arms," *The International Journal of Robotics Research*, vol. 35, no. 6, pp. 695–722, May 1, 2016.

[26] Caters Clips, director, *Hilarious Moment Elephant Steals Tour Guide's Hat*, Oct. 11, 2016.

[27] M. Kawato, "Internal models for motor control and trajectory planning," *Current opinion in neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[28] D. M. Wolpert, R. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends in Cognitive Sciences*, vol. 2, no. 9, pp. 338–347, Sep. 1998.

[29] W. M. Lai, D. Rubin, and E. Krempl, *Introduction to Continuum Mechanics*. Butterworth-Heinemann, Jul. 23, 2009, 549 pp.

[30] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer US, 2004.

[31] T. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 124–134, 2018.

[32] C. Della Santina, R. Katzschmann, A. Bicchi, and D. Rus, "Model-based dynamic feedback control of a planar soft robot: Trajectory tracking and interaction with the environment," *Int. J. Robot. Res.*, vol. 39, no. 4, pp. 490–513, 2020.

[33] R. Webster and B. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *Int. J. Rob. Res.*, vol. 29, no. 13, pp. 1661–1683, 2010.

[34] Z. Zhang, J. Dequidt, A. Kruszewski, F. Largilliere, and C. Duriez, "Kinematic modeling and observer based control of soft robot using real-time finite element method," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 5509–5514.

[35] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft Robot Control With a Learned Differentiable Model," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, May 2020, pp. 417–423.

[36] D. Q. Cao and R. W. Tucker, "Nonlinear dynamics of elastic rods using the Cosserat theory: Modelling and simulation," *International Journal of Solids and Structures*, vol. 45, no. 2, pp. 460–477, Jan. 15, 2008.

[37] D. C. Rucker and R. J. Webster III, "Statics and Dynamics of Continuum Robots With General Tendon Routing and External Loading," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1033–1044, Dec. 2011.

[38] F. Renda, F. Boyer, J. Dias, and L. Seneviratne, "Discrete Cosserat Approach for Multisection Soft Manipulator Dynamics," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1518–1533, Dec. 2018.

[39] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE Int. Conf. Robot. Autom.*, 2013, pp. 982–987.

[40] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1413–1419, 2017.

[41] E. Coevoet *et al.*, "Software toolkit for modeling, simulation, and control of soft robots," vol. 31, no. 22, pp. 1208–1224, 2017.

[42] S. E. Navarro, S. Nagels, H. Alagi, *et al.*, "A Model-Based Sensor Fusion Approach for Force and Shape Estimation in Soft Robotics," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5621–5628, 2020.

[43] W. B. J. Zimmerman, *Multiphysics Modeling With Finite Element Methods*. World Scientific Publishing Company, Oct. 25, 2006, 434 pp.

[44] H. Khalil, *Nonlinear Systems*, 3rd, Ed. Prentice Hall, 2002.

[45] Z. Gong, X. Fang, X. Chen, *et al.*, "A soft manipulator for efficient delicate grasping in shallow water: Modeling, control, and real-world experiments," *The International Journal of Robotics Research*, vol. 40, no. 1, pp. 449–469, Jan. 2021.

[46] Z. Zhang, M. Bieze, J. Dequidt, A. Kruszewski, and C. Duriez, "Visual servoing control of soft robots based on finite element model," in *IEEE Int. Conf. Intell. Robots Syst.*, 2017, pp. 2895–2901.

[47] M. Thieffry, A. Kruszewski, C. Duriez, and T.-M. Guerra, "Control design for soft robots based on reduced-order model," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 25–32, 2018.

[48] M. Thieffry, "Model-Based Dynamic Control of Soft Robots," Ph.D. dissertation, Université Polytechnique des Hauts-de-France, 2019.

[49] M. Thieffry, A. Kruszewski, T.-M. Guerra, and C. Duriez, "LPV framework for Non-Linear Dynamic Control of Soft Robots using Finite Element Model," *IFAC-PapersOnLine*, 21st IFAC World Congress, vol. 53, no. 2, pp. 7312–7318, Jan. 1, 2020.

[50] Z. Lendek, T.-M. Guerra, R. Babuska, and B. de Schutter, *Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models*. 2010.

[51] C. D. Santina and D. Rus, "Control Oriented Modeling of Soft Robots: The Polynomial Curvature Case," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 290–298, Apr. 2020.

[52] G. Fang, Y. Tian, Z.-X. Yang, J. M. P. Geraedts, and C. C. L. Wang, *Efficient Jacobian-Based Inverse Kinematics with Sim-to-Real Transfer of Soft Robots by Learning*, Jun. 5, 2022.

[53] J. Santoso and C. D. Onal, "An Origami Continuum Robot Capable of Precise Motion Through Torsionally Stiff Body and Smooth Inverse Kinematics," *Soft Robotics*, vol. 8, no. 4, pp. 371–386, Aug. 1, 2021.

[54] K.-J. Bathe, *Finite Element Procedures*. Klaus-Jurgen Bathe, 2006.

[55] K. de Payrebrune and O. O'Reilly, "On constitutive relations for a rod-based model of a pneu-net bending actuator," *Extreme Mech. Lett.*, vol. 8, pp. 38–46, 2016.

[56] J. Chenevier, D. Gonzalez, J. V. Aguado, F. Chinesta, and E. Cueto, "Reduced-order modeling of soft robots," *PloS One*, vol. 13, no. 2, 2018.

[57] J.-H. She, X. Xin, and Y. Pan, "Equivalent input disturbance approach Analysis and application to disturbance rejection in dual stage feed drive control system," *IEEE/ASME Trans. Mechatron.*, vol. 16, no. 2, pp. 330–340, 2010.

[58] Z. Wang, J. She, Z.-T. Liu, and M. Wu, "Modified equivalent input disturbance approach to improving disturbance rejection performance," *IEEE Trans. Indus. Electron.*, pp. 1–1, 2021.

[59] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*. SIAM, Philadelphia, 2005.

[60] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox, *Model Reduction and Approximation: Theory and Algorithms*. SIAM, Philadelphia, 2017.

[61] S. Prajna, "POD model reduction with stability guarantee," in *IEEE Int. Conf. Decision Control*, vol. 5, 2003, pp. 5254–5258.

[62] D. Koenig, "Unknown input proportional multiple-integral observer design for descriptor systems: Application to state and fault estimation," *IEEE Trans. Autom. Control*, vol. 50, no. 2, pp. 212–217, 2005.

[63] M. Chilali, P. Gahinet, and P. Apkarian, "Robust pole placement in LMI regions," *IEEE Trans. Autom. Control*, vol. 4, no. 12, pp. 257–270, 1999.

[64] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in Matlab," in *IEEE Int. Symp. Comput. Aided Control Syst. Des.*, Taipei, 2004, pp. 284–289.

[65] E. Coevoet, A. Escande, and C. Duriez, "Soft robots locomotion and manipulation control using FEM simulation and quadratic programming," in *IEEE Int. Conf. Soft Robot.*, 2019, pp. 739–745.

[66] A. Marchese, R. Tedrake, and D. Rus, "Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator," vol. 35, no. 8, pp. 1000–1019, 2016.

[67] A. Bajo, R. Goldman, and N. Simaan, "Configuration and joint feedback for enhanced performance of multi-segment continuum robots," in *IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2905–2912.

[68] J. Yoneyama, M. Nishikawa, H. Katayama, and A. Ichikawa, "Output stabilization of Takagi–Sugeno fuzzy systems," *Fuzzy Sets and Systems*, vol. 111, pp. 253–266, Apr. 1, 2000.

[69] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia, PA: SIAM, 1994, vol. 15.

[70] J. De Caigny, R. Pintelon, J. F. Camino, and J. Swevers, "Interpolated Modeling of LPV Systems," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2232–2246, Nov. 2014.

[71] Q. Zhang, L. Ljung, and R. Pintelon, "On Local LTI Model Coherence for LPV Interpolation," *IEEE Trans. Automat. Contr.*, vol. 65, no. 8, pp. 3671–3676, Aug. 2020.

[72] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2016.

[73] F. Schwenker, H. A. Kestler, and G. Palm, *Contributed article Three*, 2000.

[74] J. Park and I. W. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, Jun. 1991.

[75] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, Jul. 3, 2003, 271 pp.

[76] S. Vijayakumar and S. Schaal, "Locally Weighted Projection Regression : An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space," *undefined*, 2000.

[77] S. Chen, C. Cowan, and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[78] R. Toth, *Modeling and Identification of Linear Parameter-Varying Systems*. Springer, Jul. 11, 2010, 337 pp.

[79] M. Thieffry, A. Kruszewski, T.-M. Guerra, and C. Duriez, "Trajectory tracking control design for large-scale linear dynamical systems with applications to soft robotics," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 2, pp. 556–566, 2021.

[80] J. Zhang, X. Chen, P. Stegagno, and C. Yuan, "Nonlinear Dynamics Modeling and Fault Detection for a Soft Trunk Robot: An Adaptive NN-Based Approach," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7534–7541, Jul. 2022.

[81] M. Johansson, A. Rantzer, and K.-E. Arzen, "Piecewise quadratic stability of fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 713–722, Dec. 1999.

[82] H. Schulte and H. Hahn, "Fuzzy state feedback gain scheduling control of servo-pneumatic actuators," *Control Engineering Practice*, Fuzzy System Applications in Control, vol. 12, no. 5, pp. 639–650, May 1, 2004.

[83] E. Kim and S. Kim, "Stability analysis and synthesis for an affine fuzzy control system via LMI and ILMI: A continuous case," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 3, pp. 391–400, Jun. 2002.

[84] L. Rodrigues and S. Boyd, "Piecewise-affine state feedback for piecewise-affine slab systems using convex optimization," *Systems & Control Letters*, vol. 54, no. 9, pp. 835–853, Sep. 1, 2005.

[85] W.-H. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-Observer-Based Control and Related Methods—An Overview," *IEEE Trans. Ind. Electron.*, vol. 63, no. 2, pp. 1083–1095, Feb. 2016.

[86] A. Mohammadi, M. Tavakoli, H. J. Marquez, and F. Hashemzadeh, "Nonlinear disturbance observer design for robotic manipulators," *Control Engineering Practice*, vol. 21, no. 3, pp. 253–267, Mar. 1, 2013.

[87] J. Huang, S. Ri, L. Liu, Y. Wang, J. Kim, and G. Pak, "Nonlinear Disturbance Observer-Based Dynamic Surface Control of Mobile Wheeled Inverted Pendulum," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2400–2407, Nov. 2015.

[88] J. F. Guerrero-Castellanos, H. Rifaï, V. Arnez-Paniagua, J. Linares-Flores, L. Saynes-Torres, and S. Mohammed, "Robust Active Disturbance Rejection Control via Control Lyapunov Functions: Application to Actuated-Ankle–Foot-Orthosis," *Control Engineering Practice*, vol. 80, pp. 49–60, Nov. 1, 2018.

[89] A. San-Miguel, V. Puig, and G. Alenyà, "Disturbance observer-based LPV feedback control of a N-DoF robotic manipulator including compliance through gain shifting," *Control Engineering Practice*, vol. 115, p. 104 887, Oct. 1, 2021.

[90] S. H. Zak, *Systems and Control*, 1st edition. New York: Oxford University Press, Dec. 19, 2002, 720 pp.

[91] T. M. Guerra and L. Vermeiren, "LMI-based relaxed nonquadratic stabilization conditions for nonlinear systems in the Takagi–Sugeno's form," *Automatica*, vol. 40, no. 5, pp. 823–829, May 1, 2004.

[92] K. Tanaka, T. Ikeda, and H. Wang, "Fuzzy regulators and fuzzy observers: Relaxed stability conditions and LMI-based designs," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 2, pp. 250–265, May 1998.

[93] K. L. Moore, *Iterative Learning Control for Deterministic Systems*. Springer Science & Business Media, Dec. 6, 2012, 158 pp.

[94] L. Hladowski, K. Galkowski, W. Nowicka, and E. Rogers, "Repetitive process based design and experimental verification of a dynamic iterative learning control law," *Control Engineering Practice*, vol. 46, pp. 157–165, Jan. 1, 2016.

[95] L. Blanken and T. Oomen, "Multivariable Iterative Learning Control Design Procedures: From Decentralized to Centralized, Illustrated on an Industrial Printer," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1534–1541, Jul. 2020.

[96] Y. Chen, B. Chu, C. T. Freeman, and Y. Liu, "Generalized iterative learning control with mixed system constraints: A gantry robot based verification," *Control Engineering Practice*, vol. 95, p. 104 260, Feb. 1, 2020.

[97] J. D. Ratcliffe, P. L. Lewin, E. Rogers, J. J. Hatonen, and D. H. Owens, "Norm-Optimal Iterative Learning Control Applied to Gantry Robots for Automation Applications," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1303–1307, Dec. 2006.

[98] S.-K. Oh and J. M. Lee, "Iterative learning model predictive control for constrained multivariable control of batch processes," *Computers & Chemical Engineering*, vol. 93, pp. 284–292, Oct. 4, 2016.

[99] M. Norrlöf, "Iterative Learning Control : Analysis, Design, and Experiments," 2000.

[100] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer Science & Business Media, May 20, 2008, 1627 pp.

[101] R. Vinter, *Optimal Control*. Boston: Birkhäuser, 2010.

[102] J. Zhang, "Kinodynamic Motion Planning for Robotics: A Review," in *2021 5th International Conference on Robotics and Automation Sciences (ICRAS)*, Jun. 2021, pp. 75–83.

[103] H. Halalchi, E. Laroche, and G. I. Bara, "Flexible-Link Robot Control Using a Linear Parameter Varying Systems Methodology," *International Journal of Advanced Robotic Systems*, vol. 11, no. 3, p. 46, Mar. 1, 2014.

[104] G. Mercère, M. Lovera, and E. Laroche, "Identification of a flexible robot manipulator using a linear parameter-varying descriptor state-space structure," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec. 2011, pp. 818–823.

[105] F. Holsten, M. P. Engell-Norregard, S. Darkner, and K. Erleben, "Data Driven Inverse Kinematics of Soft Robots using Local Models," in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada: IEEE, May 2019, pp. 6251–6257.

[106] H. Di, Y. Xin, and J. Jian, "Review of optical fiber sensors for deformation measurement," *Optik*, vol. 168, pp. 703–713, Sep. 1, 2018.

[107] F. Xu, H. Wang, W. Chen, and Y. Miao, "Visual Servoing of a Cable-Driven Soft Robot Manipulator With Shape Feature," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4281–4288, Jul. 2021.

[108] T. Bieze, A. Kruszewski, B. Carrez, and C. Duriez, "Design, implementation, and control of a deformable manipulator robot based on a compliant spine," *Int. J. Rob. Res.*, vol. 39, no. 14, pp. 604–619, 2020.

[109] A. Preumont, *Vibration Control of Active Structures* (Solid Mechanics and Its Applications), red. by G. M. L. Gladwell. Dordrecht: Springer Netherlands, 2011, vol. 179.

[110] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-Based Model Predictive Control for Autonomous Racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[111] A. Buelta, A. Olivares, E. Staffetti, W. Aftab, and L. Mihaylova, "A Gaussian Process Iterative Learning Control for Aircraft Trajectory Tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 3962–3973, 2021.

[112] M. A. Alvarez, L. Rosasco, and N. D. Lawrence, *Kernels for Vector-Valued Functions: A Review*, Apr. 2012.

[113] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., Berlin, Heidelberg: Springer, 2004, pp. 63–71.

[114] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04, New York, NY, USA: Association for Computing Machinery, Aug. 1, 2004, pp. 522–531.

[115] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović, "Mesh-based inverse kinematics," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 488–495, Jul. 1, 2005.

[116] J. van Zundert and T. Oomen, "On inversion-based approaches for feedforward and ILC," *Mechatronics*, vol. 50, pp. 282–291, Apr. 1, 2018.

[117] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *2008 American Control Conference*, Jun. 2008, pp. 212–217.

[118] A. Yeşildirek and F. L. Lewis, "Feedback linearization using neural networks," *Automatica*, vol. 31, no. 11, pp. 1659–1664, Nov. 1, 1995.

[119] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5183–5189.

[120] T. Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft Robot.*, vol. 5, no. 2, pp. 149–163, 2018.

[121] R. S. Sutton, A. G. Barto, and C.-D. A. L. L. A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998, 356 pp.

[122] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3389–3396.

[123] S. Satheeshbabu, N. K. Uppalapati, T. Fu, and G. Krishnan, "Continuous Control of a Soft Continuum Arm using Deep Reinforcement Learning," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, May 2020, pp. 497–503.

[124] M. Capotondi, G. Turrisi, C. Gaz, V. Modugno, G. Oriolo, and A. D. Luca, "An Online Learning Procedure for Feedback Linearization Control without Torque Measurements," in *Conference on Robot Learning*, PMLR, May 12, 2020, pp. 1359–1368.

[125] J. Bolder and T. Oomen, "Rational Basis Functions in Iterative Learning Control—With Experimental Verification on a Motion System," *IEEE Trans. Contr. Syst. Technol.*, vol. 23, no. 2, pp. 722–729, Mar. 2015.

[126] R. Sanner and J.-J. Slotine, "Gaussian networks for direct adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837–863, Nov. 1992.

[127] R. Horowitz, "Learning Control for Robot Manipulators,"," *A S M E J O U R -Nal of Dynamic Systems, Measurement, and Control,*, pp. 402–411, 1993.

[128] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal Kernels," *Journal of Machine Learning Research*, vol. 7, no. 95, pp. 2651–2667, 2006.

[129] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, no. 3, pp. 337–404, 1950.

[130] S. Vijayakumar, A. D'Souza, and S. Schaal, "LWPR: A Scalable Method for Incremental Online Learning in High Dimensions," Jun. 2005.

[131] N. Lawrence, "Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data," in *Advances in Neural Information Processing Systems*, vol. 16, MIT Press, 2003.

[132] A. Kruszewski, R. Wang, and T. M. Guerra, "Nonquadratic Stabilization Conditions for a Class of Uncertain Nonlinear Discrete Time TS Fuzzy Models: A New Approach," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 606–611, Mar. 2008.

[133] J. Daafouz and J. Bernussou, "Parameter dependent Lyapunov functions for discrete time systems with time varying parametric uncertainties," *Systems & Control Letters*, vol. 43, no. 5, pp. 355–359, Aug. 15, 2001.

[134] Q. Wang and Y. Liu, "Review of optical fiber bending/curvature sensor," *Measurement*, vol. 130, pp. 161–176, Dec. 1, 2018.

[135] W. P. M. H. Heemels, J. Daafouz, and G. Millerioux, "Observer-Based Control of Discrete-Time LPV Systems With Uncertain Parameters $ $," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2130–2135, Sep. 2010.

[136] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 2592–2599.

[137] Y. Cheng, L. Sun, and M. Tomizuka, "Human-Aware Robot Task Planning Based on a Hierarchical Task Model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1136–1143, Apr. 2021.