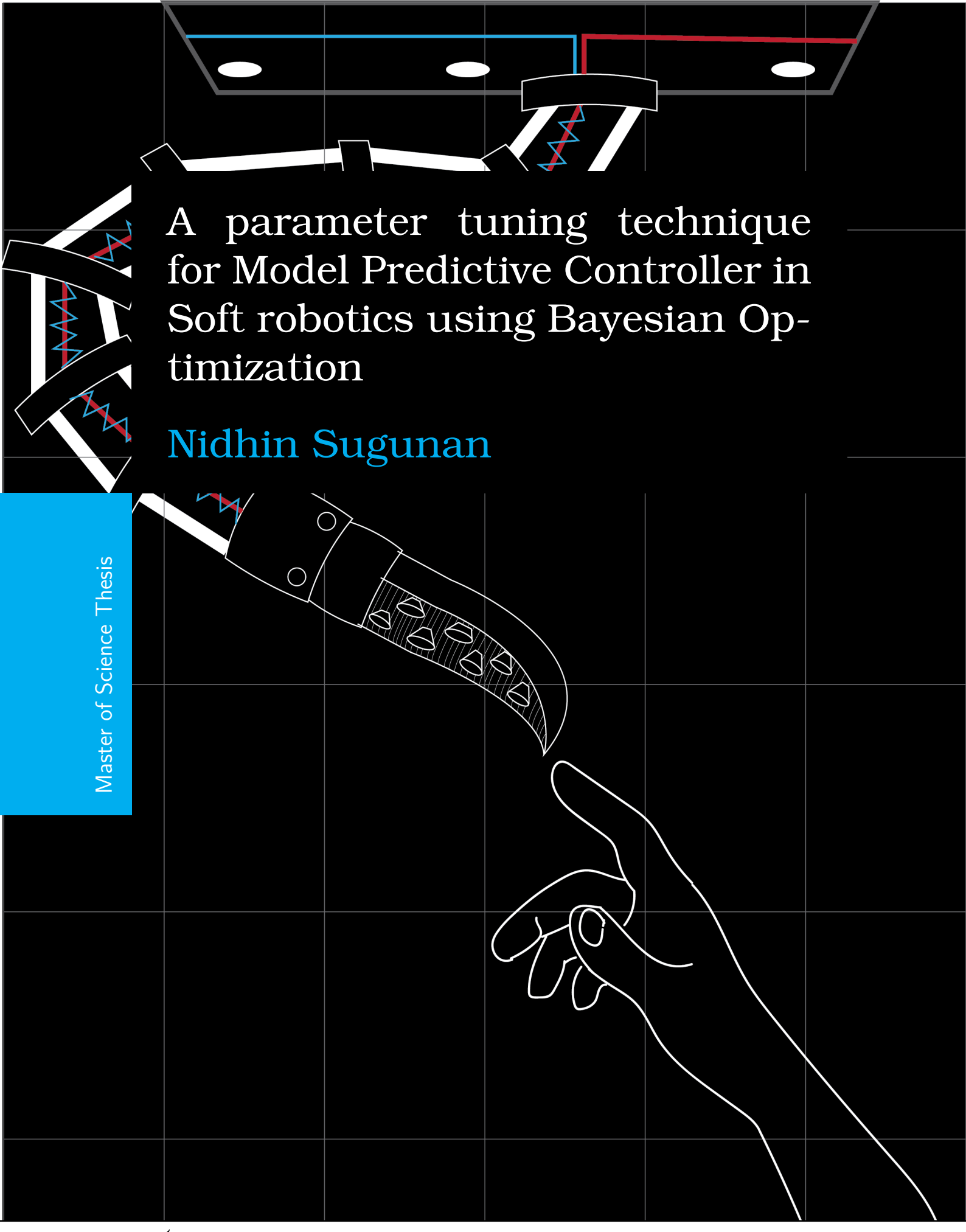
A parameter tuning technique for Model Predictive Controller in Soft robotics using Bayesian Optimization

Nidhin Sugunan

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

A parameter tuning technique for Model Predictive Controller in Soft robotics using Bayesian Optimization

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Nidhin Sugunan

August 21, 2024

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

This thesis explores a Bayesian Optimization technique for improving the tuning process of Model Predictive Control systems applied to soft robotics. Due to their high compliance and actuation redundancy, soft robotic systems are challenging to control through traditional rigid control frameworks. The objective of this study is to automate the hyperparameter tuning of MPC in order to enhance adaptability and efficiency in the control systems, handling the intricate behaviour of soft robots.

The study employs BO, leveraging its capability to efficiently navigate complex and high-dimensional optimization landscapes through a Gaussian Process (GP)-based surrogate model. This allows a systematic exploration and exploitation of the hyperparameter space toward setting up MPC optimal conditions that improve the performance metrics, such as response time and stability, under operational constraints. Two main acquisition functions, Expected Improvement (EI) and Lower Confidence Bound (LCB), are evaluated for their effectiveness in balancing exploration of the parameter space with exploitation of promising regions.

Obtained results, from simulations analyses, show that BO significantly reduces the manual effort involved in MPC tuning. The study also looks into the performance of the BO approach under varying initial conditions and introduces variance to the weights of the MPC to analyse the performance of BO under uncertainty.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Dr. Azita Dabiri for her continuous support and guidance. I would also like express my gratitude towards my academic counselors Peggy and Evert for their thoughtful advises and guidance during the difficult period.

Delft, University of Technology
August 21, 2024

Nidhin Sugunan

Chapter 1

Introduction

1-1 Soft robots

Robots come in various shapes and sizes depending upon their intended use. Their impact on the society in which we live can be characterized by their ability to improve our lives by offering faster and cheaper products with high precision. The level of competency demonstrated by the robots is highly challenging for humans to accomplish. Hence, the influence of robots has become more prominent over the years, ranging from the food industry to aerospace.

From the first robot introduced by General Motors in 1961 to the Mars Perseverance rover in recent years, remarkable development has occurred in robotics. Among the many developments in the field of robotics, soft robots have recently emerged as a prominent and noteworthy advancement. They represent a new frontier and a significant development among the latest trends and developments. With the further development of soft robots, new challenges and opportunities have established a foundation for research and development in various engineering fields.

When comparing soft robots to conventional robots, there are several separate and distinct characteristics that set them apart. Traditional robots are rigid and stiff in physical structure, mainly due to the materials used in making them. However, soft robots are made up of "soft" or flexible parts. There are several cases where the joint or moving part is made of flexible materials, which still qualify as soft robots. Traditional robots are developed to perform specific tasks in a structured environment, in contrast to soft robotics, which are developed to perform in an unstructured environment [52]. Figure 1-1 shows the distinction between traditional and soft robots based on the material used and degrees of freedom. Soft robots stand out from their traditional counterparts in that they are more flexible and adaptive, making human-machine contact safer. Hence, soft robots provide usability in various applications where accuracy, adaptivity, safety and performance are preferred [23], [65]. A brief history of the evolution of soft robots is shown in Figure 1-2

In research articles [65], [76], the robots made of soft materials are considered soft robots. In [23], a soft robot is defined as *"Robotic systems with purposefully designed compliant element*

embedded into their mechanical structure". The inspiration behind the design of soft robots has come from biological systems, exploiting their softness and flexibility to reduce the complexity of environmental interactions further. This led [76] to characterize soft robots as a new class of machines that are created from the study of biological processes. Some examples are the jellyfish [79], [34], worms robot [13], octopus arm inspired by octopus [51], [59], manta swimming robot [72], robot hydraulic autonomous soft robotic fish [56] (both motivated by fish) and so on.

Based on the inspiration from the primary source, soft robotics research is divided into two primary divisions: articulated soft robots and continuum or continuous soft robots. While

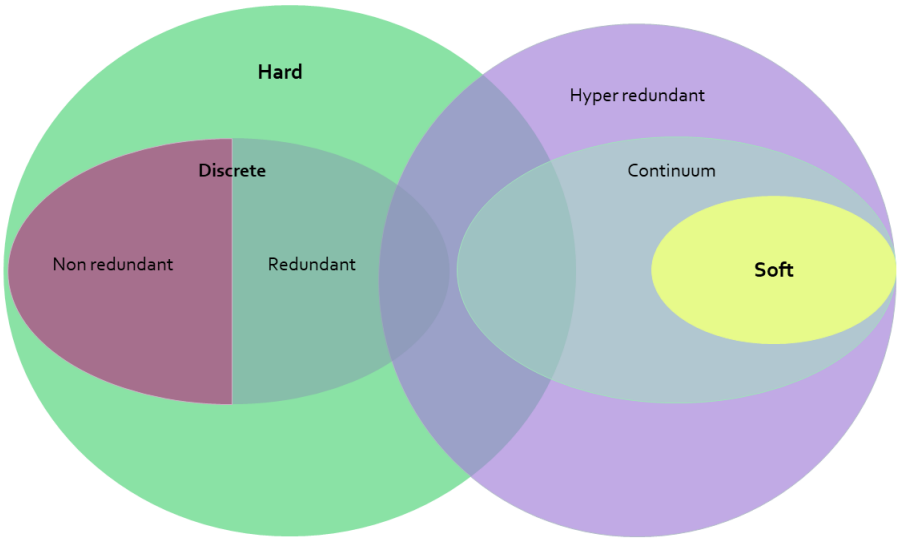


Figure 1-1: Classification of robots based on material and degrees of freedom [60]

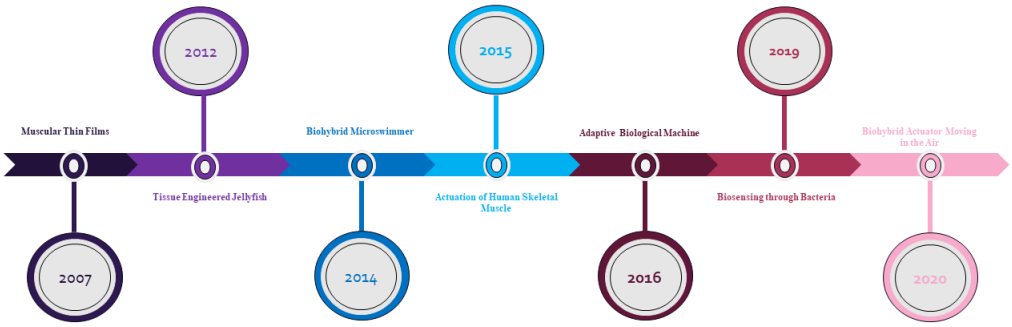


Figure 1-2: The evolution of soft robot [60]

articulated soft robot research is roused from vertebrate biological systems, continuous soft robot research is motivated by invertebrates. The continuous soft robot is also influenced by boneless parts of the animal or other parts that are able to deform continuously [23]. Furthermore, a wide variety of soft robots are described based on their modelling technique, sensing technology, actuation method, fabrication, control strategy and materials. Therefore, there are several critical challenges in the domain of soft robotics because traditional robotics relies on conventional methodologies that are either inapplicable or just partially applicable; this presents numerous vital issues in the field of soft robotics. Thus leading to comprehensive research in this domain [6].

1-2 Motivation

Soft robots are complex, and due to the uncertainties involved, many challenges in design, modelling, fabrication and control (to name a few) must be addressed before their full potential can be realised. Methodologies for controlling traditional robots were deemed a bad fit for soft robots. This is due to their nonlinear characteristics, such as soft nature, underactuation, and several degrees of freedom. Various studies have been conducted to overcome these limitations, and an overview of such challenges is mentioned below.

Soft materials have elastic properties and thus can perform several complex motions like bending, twisting, compressing, etc. Unlike traditional robots, soft robots' movement is not limited to planar motions, with several degrees of freedom added. Therefore, designing soft robots deals with several challenges. Soft materials have elastic properties and thus can perform several complex motions like bending, twisting, compressing, etc. Different mechanical components or materials are used to mimic the functions of muscles, ligaments, and tendons showing that their movements are not limited to planar motions. Pneumatic artificial muscles [77], tendon-driven soft limbs [58], cable-driven mechanisms [48] [1], smart materials like dielectric elastomers [2], and shape memory polymers [7] (both used in developing soft muscles) are a few examples.

Determining the optimal softness is a challenge in designing these robots. However, this softness makes modelling these robots more complicated and challenging. The conventional modelling techniques, like the standard ordinary differential equations for a Lagrangian system, can not be applied due to the continuous nature posed by the soft robots [23]. The soft sensors and actuators further increase the complexity due to the tolerance from manufacturing, variations in the properties of elastomers and calibration errors. Their hyperelastic nature contributes to non-linearity, wear and tear, high hysteresis, and drifts in the model. Moreover, time-varying properties (like friction and ageing) along with high degrees of freedom further limit the modelling of soft robots [65]. Thus, the mathematical modelling of a soft robot with high accuracy poses a considerable challenge and makes it more complex to control these robots to perform tasks.

Model-based and data-driven techniques are widely used when it comes to modelling traditional robots. Soft robots have different methods, but these two modelling techniques are widely practised. Data-driven techniques were the first techniques to be used in developing the models, as the model-based techniques were intricate [24]. Further, different assumptions and approximations led to decreasing complexity in using model-based techniques and

methods like piecewise constant strain approximations [21], rod model [37], energy-based model [12] and finite element model [86] were delivered, which paved the way for having fairly accurate models for the control design process. These models had a lot of advantages over their predecessors, and an elaborated discussion of this is made in Chapter 2.

With the availability of a reasonably accurate model, several control strategies are available now, from open-loop control to model-based reinforcement learning [43]. Model Predictive Controller (MPC), is a promising approach of model-based controller for its efficient control and good trajectory tracking performance. This controller can minimize a cost function subject to the system's dynamics as constraints over a finite time horizon. According to different research, MPC's characteristic of predicting the future over a prediction horizon helps the controller perform well against the soft nature of the robot [38], [10], [16]. If cost is not a concern, MPC is one of the best control strategies for developing model-based controllers. However, manually tuning the performance parameter is time-consuming, can cause instability and could even damage the soft robot [61].

1-3 Problem statement

Trial and error is the traditional way of tuning model-based controllers. The careful tuning, however, does not guarantee the best results. To address this challenge, there are several automated tuning techniques available in the literature. Gradient-based auto-tune framework [20] establishes that manual tuning is far inferior to model-based controllers' automatic tuning. This technique has shown promising results even for the likes of controllers like LQR [57]. Reinforcement learning is used in tuning the MPC [11]. AutoMPC, an open-source library in Python, is used for tuning data-driven MPC [28], [63], thus establishing the relevance.

The goal of the thesis is to get closer to solving the challenges that can be decomposed as:

- Improve the performance of MPC, hence overcoming the limiting model accuracy issue in soft robots.
- Replace the manual trial and error practice with a less time-consuming and exhaustive approach followed in MPC.

While attempting to address two distinct issues, it becomes evident that these problems are interconnected. The thesis will take a step towards solving this challenge in a simplified framework. Therefore, the challenge that is posed and formulated by the problem statement is as follows:

> How do we find an optimal trade-off (according to a performance metric) between exhaustive search and incorporating process knowledge in tuning an Model Predictive Controller (MPC) for soft robots?

One solution to this question is using Bayesian optimization Bayesian Optimization (BO) to auto-tune the performance parameter of MPC. BO is a proven tool as a global optimization algorithm [31], [45]. BO provides a choice as it is used in many other scenarios to improve the performance of MPC [54]. Therefore, in the context of soft robots, a method is developed primarily viewed from an analytical perspective on how the MPC performs.

1-4 Thesis outline

The organizational structure of the thesis is as follows: **Chapter 2** includes brief literature on different modelling and control techniques used in soft robots and the choice of model and controller used for this work. **Chapter 3** overviews the concepts involved in Bayesian optimization and the working principle. **Chapter 4** provides results of the experiments conducted, performance analysis and the conclusion on the overall contribution.

Chapter 2

Soft robot modelling and control

The modelling of soft robots is one of the challenges in realizing a soft robot. Various types of modelling are available, and the review is done first, closing with prevalent control strategies used for soft robots. Further in this chapter, a model is chosen, and its dynamics are discussed. The second half of the chapter includes the controller selected for this thesis, its motivation, and the design of the controller.

2-1 Related work

In this section, first, the commonly used models and their characteristics are discussed. Also, looking into the different types of modelling techniques used, as well as their advantages and drawbacks. Later, this section delves into previous work done to tackle the soft robot control challenges.

The complex dynamics of soft robots are broadly tackled in two distinct ways: model base and model-free, which will be elaborated on below with relevant references. A brief of both techniques will be discussed in this section. Due to the extensiveness of this particular research, only the popular methods are mentioned. Additionally, several survey papers [24], [66], [50], [76], [3], [4] are available that provide a more in-depth analysis of the different modelling techniques used.

Modelling methods

The complex of soft robots made the modelling challenging, especially since model-based methods were limited. These limitations were mainly due to the intensity and difficulty of solving the excessively intricate models of soft robots. However, various new modelling techniques emerged over time [24]. Another reason for this possibility was the different approximation techniques and, in some cases, assumptions that made the models simpler without compromising a lot on their main characteristics. One such modelling method is Constant

Curvature (CC) [82], one of the primarily used kinematic models initially employed in modelling the continuum soft robots. This method assumes that the soft robot comprises constant curvature segments, each represented by a reference frame attached to its end. The configuration of each segment can be fully defined by the reference frames, allowing the robot's kinematics to be described by homogeneous transformations mapping each reference system to the subsequent one. This modelling technique is easy and simplified. However, when external force is applied, the model fails to capture the entirety of the complex behaviour of the soft robot [66]. While this model is a primitive method, it displays lower accuracy than other available techniques but is good enough to integrate the essential characteristics of the soft robot.

Piecewise Constant Curvature (PCC) [25] also works on the same assumption that the soft robot is formed of constant curvature and provides a homogeneous transformation for each segment using geometrical relations. The soft robot is represented as a series of finite arcs defined by the bending plane, radius and angle of the arc. This model incorporates elastic and dissipative terms to model thus the behaviour of the soft robot is not lost. The elasticity of a link is modelled through a continuous distribution of infinitesimal springs along the cross-sectional area of a segment, and the damping is introduced through linear dampers parallel to the springs. These elements contribute to the overall impedance of the soft robot. Therefore, it helps develop control strategies to perform dynamic tasks like reference tracking in an unstructured environment, as shown in [25] and [43], where, in the latter, an MPC is used, and a nonlinear controller is used in the former. Nevertheless, in [19] claims a few drawbacks that are associated with the PCC model. First, the impact of friction and other mechanical implications are not factored in PCC. Second are the large nonlinearity characteristics and the numerical instability at singular points. The authors claim to have overcome the numerical instability with an improved Approximate Piecewise Constant Curvature (APCC) method. Another limitation of PCC was that the assumed model was considered in planar, which limited its relevance in 3-D as the real-world robots are three-dimensional. This issue is handled using PCC with state parameterization [22]. But the new approach increased the complexity since it became more intensive computationally.

A significant challenge limiting the growth of model-based methods was the complexity of the infinite dimension formulation of state space and its practicality in developing control strategies. However, developing simplified finite set variable models helped overcome the limitations. The Finite Element Method (FEM) [86] follows such a technique where kinematic is defined using the real-time FEM. Even though this method provides a better model than PCC, FEM has its shortcoming in its model. One reason for this issue is the need to specify the properties of the material of the soft robot that needs to be defined. However, the biggest problem with this method is its high computational cost; also, expanding to a dynamics model using FEM will further increase the complexity. Using FEM is difficult when developing a close-loop control due to the large dimensionality. A reduced-order model [75] was introduced to resolve this issue where the nonlinear model is reduced using Proper Orthogonal Decomposition (POD). The model is then linearized to develop a closed-loop controller. The limitations of this method come from the errors associated with reducing the model. Also, the performance is evaluated around an equilibrium point. Due to the complex nonlinear behaviour of the soft robot, this might not be desirable.

In Cosserat's theory [14], the infinite degrees of freedom property of the soft robot is defined as micro solid infinitesimal. They are then stacked together in order to maintain hyper-

redundancy. In recent research, the discrete Cosserat approach [64] was developed, where the limited state-space dimension property of the PCC is used with the Cosserat theory. Compared to all the other model-based methods, the Cosserat modelling technique provides a highly accurate model but at a cost. These methods are complex to compute and computationally expensive. This limits the ability to develop control strategies for these models [66]. The survey [24] provides in-depth information on more model-based methods used for soft robots and states that this type of method performs better than the model-free one. However, dynamic challenges like interaction with the environment, uncertainty pertaining to system parameters, and completing tasks (dynamic) are still being explored and have their shortcomings.

Data-driven methods are constructed from data collected by exciting the soft robot, hence defining their model based on observations. Collecting data in the robot's operating range can provide good accuracy since it can also map the soft behaviour. Unlike model-based techniques, no structural assumptions or approximations are made. System identification is one of the techniques used in identifying nonlinear models. In [17], Koopman operator theory is used as a system identification method to develop a nonlinear model of a soft robot. Later, the linear model is generated using the Koopman operator (a linear operator). This research claims that this is an easy technique to construct an accurate model of a soft robot. However, this technique requires the effort of the parameters to be manually tuned, which is a limitation as it involves the issues inherited from trial and error. Another problem is that imperfect tuning can result in a poor model.

Since both model-based and model-free techniques have considerable drawbacks, there have been efforts to use the first principle model to leverage the data-driven model primarily deployed in the control loop. Here, an online learning loop is established in the controller feedback. Iterative Learning Control (ILC) is a type of such learning loop. A particular task is repeated multiple times, and the loop tries to improve this task as each iteration progresses. This helps the soft robot perform dynamic tasks and perform well under uncertainty. However, it might not be suitable for soft robots with continuum robots [24].

To conclude, though extensive work has been done to obtain accurate models for soft robots, much more has to be done. Several issues need addressing, like underactuation, manoeuvring through an unconstrained environment, etc. However, these require more intensive work to solve it. Considering the challenges of modelling soft robots, we must make compromises. However, heavy simplifications reduce the overall performance, and if a simple model is to be correctly calibrated, it would lead to better results. Is obtaining a good performance with a simple model possible? However, manual trial and error is tedious. One particular improvement could be suggested: automating the tuning process using some techniques. This process seems essential since parameter tuning significantly affects the system's performance, as discussed before.

Control strategies

An essential aspect of a soft robot is its ability to perform tasks in an unstructured environment by retaining stiffness and adaptability, in which the controllers play an important role. However, there are several challenges to be faced when compared to controlling a traditional

robot. Some of the mentions are the infinite degrees of freedom under actuation and restriction of high-frequency control due to hysteresis. The soft robot control domain research is still active, and rich literature can be found in [35], [38] and [80]. The control methods are available in two types; the first is model-based control, in which the controller is developed with a physics-based model. The second type is model-free control, and the model used for developing the controller will be data-driven.

In the model-based methods, some controllers are kinematic based, where the dynamics of the soft robot are not considered, and then there is also dynamic-based, which is more complicated than kinematic to develop [38]. Kinematic models were the first to be used due to the complexity involved in developing the soft robot and are also widely used. The PCC and Cosserat rod-based models were used. In the model-free method, the Gaussian Process, online learning techniques (where the Jacobian is estimated), neural network and other data-driven approaches were used to derive the model for the kinematic controller. While the PCC and Cosserat rods were complex to compute and specific to design, they proved more accurate and reliable. Also, they provided the proof for stability analysis, which was impossible with a model-free control approach. Having to sample extensive data and the noise from the sensors made it even more challenging to achieve the desired performance in a model-free approach. One advantage of the model-free approach was that it was a convenient approach to estimate the parameters of the soft robot regardless of the extensive knowledge of the soft robot's physicality. Since the dynamics of the soft robot are not considered, the kinematic approach was limited in terms of performance [35], [38].

In the dynamic approach, the design cost is higher than the kinematic but is faster and more accurate [38]. There are several model-based dynamic approaches like PD controller [49], nonlinear controller for the PCC model [26] [25], robust control [53] and port Hamiltonian [12]. In the model-free approach, machine learning, reinforcement learning, neural networks and more based controllers have been designed, and they have drawbacks, as stated before [50]. A new approach of the model-based controller using data-driven approaches has also been designed [17], [47]. MPC is more computationally expensive than the rest of the model-based method. This is because the optimization problem is solved in every control step to obtain the optimal control output. The advantages of the MPC are the easy handling of multiple inputs without any simplifications or assumptions and the efficient use of these multiple inputs for the objective. Also, with additional constraints, MPC can overcome the actuator limits and perform well in an unstructured environment. MPC promises excellent performance, and this is also why it is a good/better choice if sufficient computation resources are spent for a good model. Therefore, MPC is deemed as a reliable and accurate control strategy with the right choice of performance parameter, increase of computation capacity and prediction horizon.

Next are state estimation techniques like the Extended Kalman Filter (EKF), where the closed-loop state estimation is performed using observers. Here, the data is acquired directly from the sensors. This method, however, has a lot of shortcomings, as the measured data contains noise and characteristics associated with the softness of the robot are not defined accurately [73]. A comprehensive study of machine-learning techniques is available in [50]. Though this data-driven technique delivers fairly accurate models, they require a large data set. This considerably increases the computation time and resources. This argument is valid for Neural Networks and Reinforcement learning techniques, and the parameter tuning required for these techniques(which affects the quality of results) is done by trial and error, which has its own shortcomings. When direct learning techniques are applied to closed-loop

control systems, they have encountered limited performance and stability issues [24]. Moreover, data-driven models are involved in modelling nonlinear characteristics of soft robots, resulting in solving non-convex optimization problems. Therefore, this method cannot assure the global minima/maxima (convergence) [50].

2-2 Spherical soft robotic arm model

For this thesis, a toy model is selected with the dynamics of a spherical soft robot in [42]. To test the auto-tuning algorithm for the MPC, it was necessary to start with a simple model, and this was found to be apt. The spherical soft robot (from here onwards, referred to as a robotic arm) is a robot made of two links (hence referred to as an arm) that is inflatable using three bellow type actuators and are connected by a soft ball-socket flexible joint as shown in Figure 2-1. This joint is restricted to two degrees of freedom, and the tip contains a motion capture sensor with a 3-D printed net as an end effector. The links of the robotic arm are made of fabric, and further details regarding its design and fabrication are provided in [41].

2-2-1 Robotic arm dynamics

In [42], the robotic arm is linearized from the arm dynamics equation Equation 2-1 and pressure dynamics equation Equation 2-2 given below:

$$\begin{aligned} \ddot{\alpha} &= -k_\alpha \alpha - d_\alpha \dot{\alpha} + h_\alpha \Delta p_\alpha \\ \ddot{\beta} &= -k_\beta \beta - d_\beta \dot{\beta} + h_\beta \Delta p_\beta \end{aligned} \tag{2-1}$$

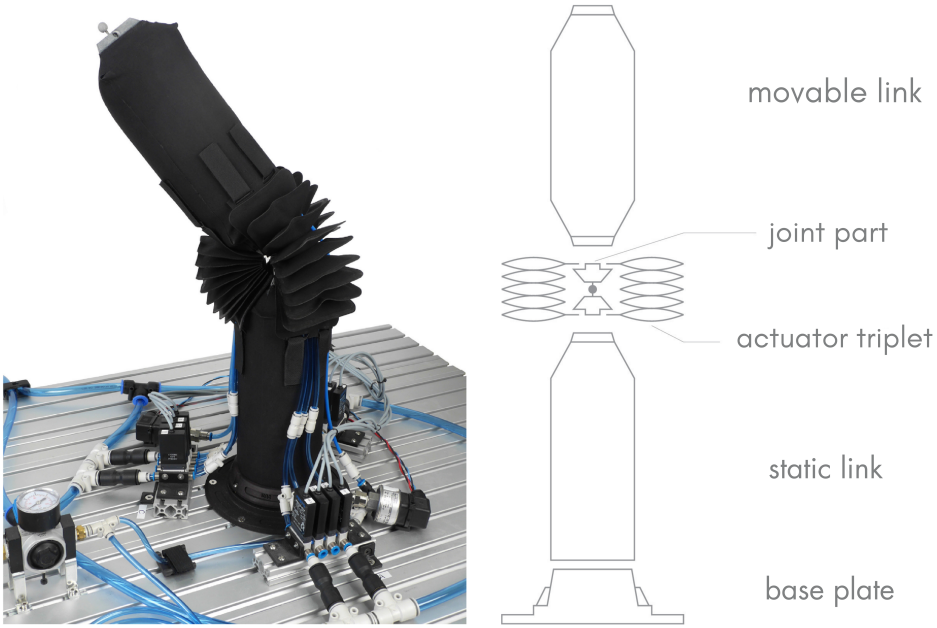


Figure 2-1: The left side is the physical structure of the spherical soft robot [42], and the right is the breakdown version with components [41].

$$\begin{aligned}\Delta\dot{p}_\alpha &= 1/\tau_\alpha \left(\Delta p_{\alpha_{ref}} - \Delta p_\alpha\right) + c_\alpha \dot{\alpha} \\ \Delta\dot{p}_\beta &= 1/\tau_\beta \left(\Delta p_{\beta_{ref}} - \Delta p_\beta\right) + c_\beta \dot{\beta}.\end{aligned} \tag{2-2}$$

The extrinsic Euler angles configuration is shown in Figure 2-2; all the other notations are defined in Table 2-1. Then, the author used linear regression to identify the parameters. Unfortunately, these parameter values were unavailable; hence, values required for this thesis are approximated from limited data available from [41]. The resultant 6 state LTI system can be represented in the form:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t),\end{aligned} \tag{2-3}$$

with starting point $x_0 = x(t_0)$. The state matrix $A \in \mathbb{R}^{6\times6}$ with state dynamics $x \in \mathbb{R}^{6\times1}$ and the input matrix $B \in \mathbb{R}^{6\times2}$ with input dynamics $u \in \mathbb{R}^{2\times1}$ is given as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 & & & \\ -k_\alpha & -d_\alpha & h_\alpha & & & \\ 0 & c_\alpha & \frac{-1}{\tau_\alpha} & & & \\ & & & 0 & 1 & 0 \\ & & & -k_\beta & -d_\beta & h_\beta \\ & & & 0 & c_\beta & \frac{-1}{\tau_\beta} \end{bmatrix}, B = \begin{bmatrix} 0 & \\ 0 & \\ \frac{-1}{\tau_\alpha} & \\ & 0 \\ & 0 \\ & \frac{-1}{\tau_\beta} \end{bmatrix} \tag{2-4}$$

$$\begin{aligned} x &= (\alpha, \dot{\alpha}, \Delta p_\alpha, \beta, \dot{\beta}, \Delta p_\beta) \\ u &= (\Delta p_{\alpha_{ref}}, \Delta p_{\beta_{ref}}). \end{aligned} \tag{2-5}$$

The definition of each parameter is given in Table 2-1. The modelling decisions taken to describe the soft robot's behaviour have limitations. However, MPC and BO are computationally expensive, so beginning with a simple, less computationally expensive model was necessary. Also, the framework presented in this thesis is analyzed academically; a linear MPC was taken to shorten the simulation time by solving a convex problem, which is also valid for solving BO. Nonlinear dynamics will drastically increase solving time. Furthermore, a BO with nonlinear constraints is required for which high-end systems are needed to solve it. Since dealing with complexity is not at the core of the problem the thesis is trying to solve; this complexity is being discarded.

2-2-2 Experiment setup of the soft robot

When designing a controller, the mathematically derived model is far from ideal, as all the system dynamics can only be incorporated partially. To distinguish the derived model from the real robot, in this thesis, the derived model will be referred to as the model, and the actual system will be referred to as the plant. The plant for this thesis is developed in `MATLAB` by varying the model parameters $\pm20\%$ using the `randn` command since the scope of the thesis is extended only to simulations and the unavailability of the dynamics of the physical ball-catching soft robot (plant).

All real-time systems (soft robots included) are in the continuous time domain. However, MPC requires a discrete-time model to function. Therefore, a discrete-time model needs to be constructed from the continuous-time model from Equation 2-3. A Zero-order Hold (ZOH) method is used to discretize Equation 2-3, and its illustration is shown in Figure 2-5. The resultant expression is given in the form:

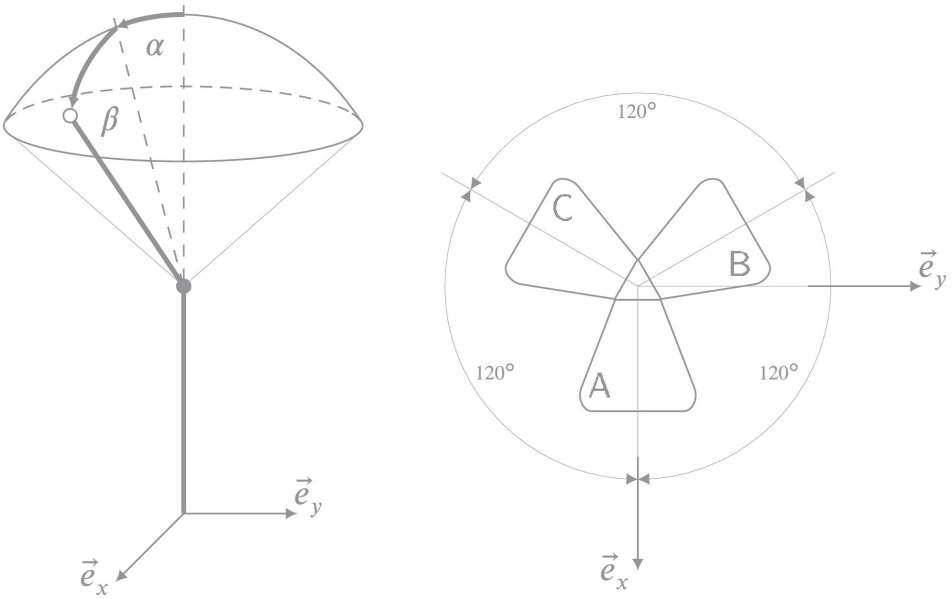


Figure 2-2: The left side represents the robotic arm orientation, and the right side shows the ball-socket joint arrangement controller by three actuators [41].

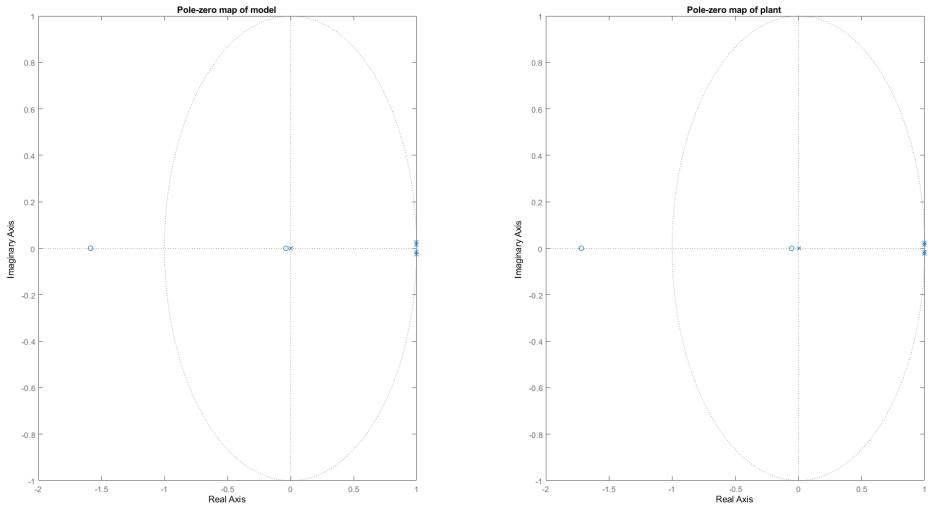


Figure 2-3: Left plot represents the pole-zero map of the model and right plot represents the plant

Notation	Definition
α	Extrinsic Euler angle
β	Extrinsic Euler angle
k_α	Stiffness coefficient w.r.t α
k_β	Stiffness coefficient w.r.t β
d_α	Damping coefficient w.r.t α
d_β	Damping coefficient w.r.t β
h_α	coefficient mapping pressure differences to angular excitations w.r.t α
h_β	coefficient mapping pressure differences to angular excitations w.r.t β
c_α	Arm movement interaction coefficient w.r.t α
c_β	Arm movement interaction coefficient w.r.t β
τ_α	closed-loop pressure dynamics time constant w.r.t α
τ_β	closed-loop pressure dynamics time constant w.r.t β
p_A	Pressure from actuator A
p_B	Pressure from actuator B
p_C	Pressure from actuator C
Δp_α	Pressure difference between p_A and p_B
Δp_α	Pressure difference between p_B and p_C
$\Delta p_{\alpha_{ref}}$	Input pressure required to reach the reference point
$\Delta p_{\beta_{ref}}$	Input pressure required to reach the reference point

Table 2-1: Table of notation used in modelling the robotic arm.

$$\begin{aligned} x^+ &= \tilde{A}x + \tilde{B}u \\ y &= Cx, \end{aligned} \tag{2-6}$$

where,

$$\tilde{A} = e^{AT_s}, \in \mathbb{R}^{6\times 6}$$
$$\tilde{B} = \int_0^{Ts} e^{AT_s} B ds, \in \mathbb{R}^{6\times 2}.$$

The rule of thumb is to take 1/10 of the rise time, which is considered to be good, and $T_s << 1$ is generally opted for fast process applications like robotics and aerospace, where T_s denotes the sampling time. For convenience, $x(k+1)$ is denoted as x^+ and $x(k)$, $u(k)$, $y(k)$ is denoted as x, u, y.

The model and plant are discretized via the Zero-Order Hold method as shown in Figure 2-5 using the `MATLAB` command `c2d`. The sampling time used for the discretization is $T_s = 0.02\,\text{s}$. Theoretically, sampling time must be 1/10 of the system's rise time as it ensures a better

representation of the system's dynamics. However, for implementation, 0.02 s is opted for several practical reasons. The soft robots usually exhibit high-frequency dynamics, and the opted T_s value can capture these dynamics and avoid computational efficiency. A smaller sampling time can provide better resolution when capturing system dynamics. In [42], the author opts for the same T_s, further supporting the choice. The pole-zero map of the model and plant is shown in Figure 2-3. The pole are either in or very close to the unit circle. Therefore, the soft robot cant be considered as strictly stable however, it can be concluded that the marginally stable since the output does not grow and explode.

Next, the step-response is plotted to inspect the model's characteristics further. Observation made from Figure 2-4 confirms that the robot exhibits an underdamped behavior. This behavior could be due to the elastic and compliant nature of the soft robot. From the point of stability, the system can be concluded as stable. The system can be concluded as stable since no unbound oscillations were observed. The A matrix is a full-rank matrix and controllable. This property enables the system to start at any point and reach a desired state in a finite time. The model assumptions include neglecting certain nonlinear effects and simplifying the dynamics to achieve computational simplicity. Thus this is one of the limitations of the model.

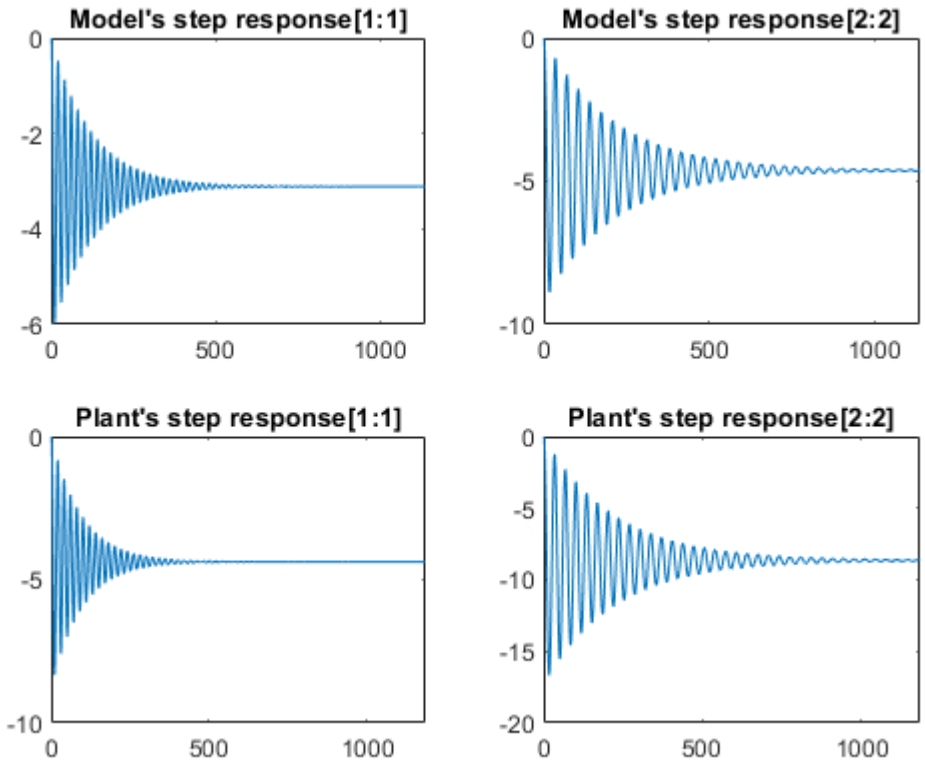


Figure 2-4: Step response of the soft robot

2-3 Soft robot control

This section will look into the formulation of the controller, which includes defining the constraints and optimization problem formulation used in this work. Furthermore, the motivation and impacts of auto-tune for MPC are examined.

2-3-1 MPC formulation

MPC is a model-based control approach utilizing optimization and control methods to be iteratively applied in order to identify and implement the best possible feedback law. This is best illustrated with an example, where the benefit of using MPC stems from its ability to predict how a system will behave in the future utilizing its dynamical model. A flexible iterative model-predictive-control (MPC) mechanism that can real-time customize its control policy for highly coupled dynamics and constrained uncertain systems [74]. Despite of those, MPC can optimize future control actions and therefore is substantial to be applied in many engineering problems such as soft robotics [61], [16].

The main components for formulating an MPC are the inequality state and input constraints, control and prediction horizon, cost functions and weight matrices. The inequality constraints of states and input are represented by an Upper Bound (UB) and Lower Bound (LB). The advantage of these constraints is to limit the input resource, mathematically define the physical limits of the soft robots and aid to a certain for the output not to blow up. The general expression for the inequality constraint of the state and input can be defined as:

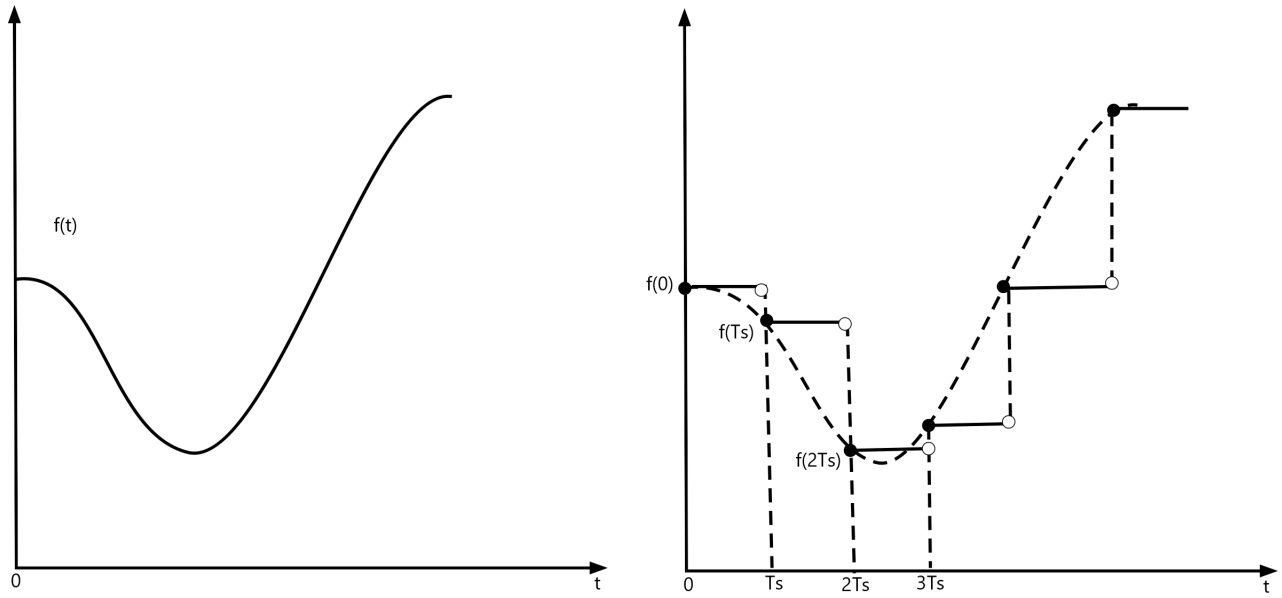


Figure 2-5: The LHS represents the continuous-time domain function, and RHS shows its zero-order hold sampling method [85]

$$x_{\min} \leq x(k) \leq x_{\max} \tag{2-7}$$
$$u_{\min} \leq u(k) \leq u_{\max}, \tag{2-8}$$

where, $x_{\min}$, $u_{\min}$ denotes the lower bound and $x_{\max}$ and $u_{\max}$ denotes the upper bound for the states and inputs respectively. The Equation 2-7 and Equation 2-8 can be represented in a compact manner such that it obeys these equations. This compact form is given as:

$$F_1 x(k) \leq c_1 \tag{2-9}$$
$$F_2 u(k) \leq c_2, \tag{2-10}$$

where,

$$F_1 = \begin{bmatrix} I_6 & 0_6 \\ 0_6 & -I_6 \end{bmatrix}, c_1 \in \mathbb{R}^{12\times 1}$$
$$F_2 = \begin{bmatrix} I_2 & 0_2 \\ 0_2 & -I_2 \end{bmatrix}, c_2 \in \mathbb{R}^{4\times 1}.$$

where, c_1 and c_2 collect the upper bound and lower bound of the inequalities in Equation 2-7, Equation 2-8. The values for the UB and LB are chosen based on the understanding of the physical structure of the robot and its limitations.

For a sampling sequence k, if M is a set of control inputs $u(k+i-1)$ for $i = 1, 2, \ldots, M$, which are calculated such that N is a set of predicted output $\tilde{y}(k+i)$ for $i = 1, 2, \ldots, N$ then the controller obtains the input sequence starting from step k optimally according to the objective function and constraints. The number of control steps M and prediction steps N is the control horizon (M) and prediction horizon (N), respectively [69]. While the control horizon optimizes the control inputs, the prediction horizon greatly impacts the performance and computation cost. A short prediction horizon gives inaccuracy in performance; however least cost and visa versa. Therefore, the choice of the prediction horizon must be done effectively.

The main objective of the controller is to reach a reference point via the soft robot. This goal can be mathematically expressed using a cost function that must be minimized. The expression is given as:

$$l(x_k, u_k) = \|x_k - x_{ref}\|_Q^2 + \|u_k - u_{ref}\|_R^2. \tag{2-11}$$

The indices for the state and input references have been omitted for clarity since they are kept constant throughout the experiments.

The cost function has a direct impact on the performance of the MPC. The weight matrices Q and R penalizes the state and input matrices respectively. Therefore, the choice made for Q and R is crucial for the MPC's performance. The weight matrices are given as:

$$Q = \begin{bmatrix} Q_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_6 \end{bmatrix} R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \tag{2-12}$$

The tuning of Q intensively impacts the steady-state response and the system's transient response. The R can impact the velocity, avoiding fast movements and acting as a constraint on it. This cost matrix also limits the undesirable input that may lead to instability [69]. By setting the inequality constraints, prediction horizon, control horizon, cost function and cost matrices, the Optimal Control Problem (OCP), which is the optimization problem, is expressed as:

$$\begin{aligned}
\underset{\mathbf{x},\,\mathbf{u}}{\text{minimize}} \quad & J_N(x,u) = \sum_{k=0}^{N-1} l(x(k),u(k)) && \text{(2-13a)}\\
\text{subject to} \quad & x^+ = \tilde{A}x(k) + \tilde{B}u(k), \quad \forall k = 0,1,\ldots,N-1, && \text{(2-13b)}\\
& x(0) = x_0, && \text{(2-13c)}\\
& F_1 x(k) \leq c_1, \quad \forall k \in [0,N], && \text{(2-13d)}\\
& F_2 u(k) \leq c_2, \quad \forall k \in [0,N-1]. && \text{(2-13e)}
\end{aligned}$$

Note that $\mathbf{x}$ and $\mathbf{u}$ are the vectors that contain the states and the inputs for the N-step horizon. The performance of the MPC depends on the model accuracy and choice of values taken for prediction horizon. Moreover, tuning the Q and R is vital since quadratic penalizing significantly effects the states and inputs of the MPC more than the rest. This effect has a critical impact on the MPC's performance.

2-3-2 Experiment setup of MPC

The MPC designed for this study is central to optimize the control performance of the soft robot. The MPC was implemented using the CasADi framework. The first step to this is to define the control objectives. The main control objective is to control the actuation inputs of the soft robot to the target angles using MPC. The OCP formulation and model are already defined in Equation 2-13a. However, there are a few more parameters that need to be defined. The values opted for these parameters have the potential to vary significantly the performance of the MPC. The constraints of the MPC like the LB and UB of the input and state constraints are defined are defined to ensure the robot operates within safe and effective operational bounds as shown in Table 2-2 by taking motivation from the values used [42]. State constraints prevent the robot from reaching physically implausible positions, while input constraints limit the maximum and minimum pressures that can be applied by the actuators. These constraints help in maintaining the structural integrity of the robot and in avoiding scenarios that could lead any failures.

The state weight matrix Q penalizes the states α and β. This will make sure that the error is minimised thus attaining the desired state. The input weight matrix R is tuned to minimize

State constraints		Values
α	LB	$-\pi$ radians
	UB	π radians
$\dot{\alpha}$	LB	-2π radians/second
	UB	2π radians/second
$\Delta p_{\alpha_{ref}}$	LB	-50 bar
	UB	50 bar
β	LB	$-\pi$ radians
	UB	π radians
$\dot{\beta}$	LB	-2π radians/second
	UB	2π radians/second
$\Delta p_{\beta_{ref}}$	LB	-50 bar
	UB	50 bar

Table 2-2: Upper and lower constraint limits used for MPC implementation

the actuation effort, which balances performance and energy efficiency. These weights are chosen as hyperparameters, which are optimized using Bayesian optimization to find the best trade-off between control accuracy and effort. The MPC problem is formulated as a Quadratic Programming (QP) problem, where the objective is to minimize the cost function subject to the constraints of the soft robot. An interior point solver is used to solve this problem in MATLAB. The solver settings are finely tuned to balance the performance and time taken to solve the control problem.

The block diagram provided illustrated in Figure 2-6 the general structure of the implemented MPC. The initial states x_0 is given to the plant. All the state measurement are assumed to be known hence the states from the plant is directly fed to the controller. The Soft Robot (model) block represents the mathematical model of the soft robot used by the MPC for predictions accurately describing the dynamics to simulate. Objective Function and Constraints are essential components of the MPC. The objective function quantifies the cost of deviations from the target and possibly control effort. The constraints ensure that the solution stays within safe and feasible operational limits. OCP block solves the corresponding optimization problem, taking the error, objective function, and constraints to compute the optimal control actions u. The control signal u is shown to act on the Real Soft Robot (plant) in the diagram.

2-3-3 Auto-tuning MPC

Model Predictive Controller (MPC) is a widely used control algorithm applied to the highly nonlinear dynamics of soft robots. Its ability to predict future states and inclusion of constraints helps achieve successful control of complex and multi-DOF systems. The performance of MPC depends greatly on the fine-tuning of its parameters, particularly the weights Q and R in the cost function to balance between state tracking errors as well as control efforts.

Working with these parameters manually can be cumbersome thus requiring a lot of trial and error to achieve desired performance. This is compounded by the non-intuitive relationships between the tuning parameters and closed-loop performance that prevent manual or heuristic approaches from working well, if at all. Currently, there are optimization-based [67], [84], [32] and machine learning-based [44] tuning approaches. However, the BO method opts for this work as it is an effective and practical approach to optimizing performance criteria [63],[29] [5], [61]. The automated tuning approach by Bayesian Optimization (BO), which is seen drastically reducing that process. BO is a powerful technique for optimizing computationally expensive black-box functions that have been successfully applied to the tuning of MPC controllers in complex systems [62], [54], [36].

There are a few advantages of auto-tuning MPC using Bayesian Optimization.

- As BO is used to systematically explore and exploit the parameter space, it can identify which set of parameters will further improve the closed-loop performance of the MPC approach ultimately helping in achieving more accurate control for out soft robot.
- BO efficiently searches for optimal parameters with its model rather than performing much manual tuning or large-scale simulations. This can greatly reduce the amount of time and computational resources needed for tuning.
- Due to the iterative nature of BO, it is possible to keep on updating the design based on real-time data and make sure that the control system adapts well in case dynamics or operating conditions change for the robot.
- Since BO-tuned MPC is able to directly minimize a performance index which considers uncertainties and disturbances by tuning on the parameters, reaching higher robust control levels than manually tuned controllers can be possible with this method.

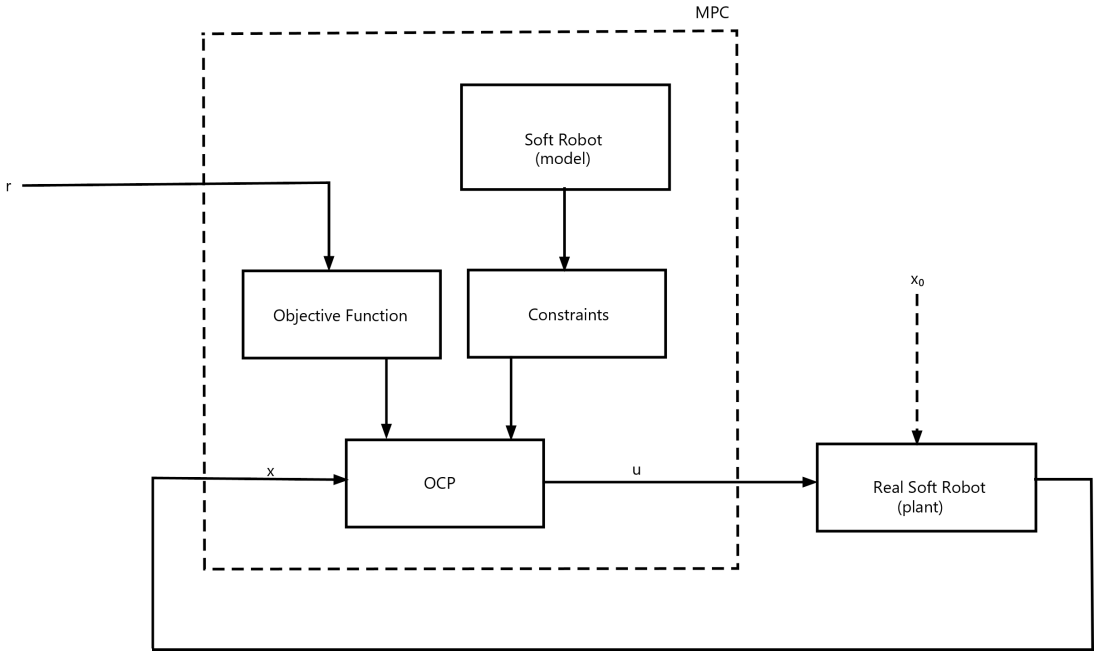


Figure 2-6: Block diagram of the MPC

In this thesis work, only the 4 hyperparameters are used for tuning. Using more than 4 has practical issues as hyperparameter tuning space is large, exceeding the hardware limits. The hyperparameter choices are Q_1, Q_4 from the Q cost matrix and R_1, R_2 from the R cost matrix given in Equation 2-12. Since a goal-oriented approach is opted, the weights, i.e. Q_1 and Q_4 corresponding to α and β, are chosen as they make the most impact and reduce the position error, are prioritized. Similarly, the remaining two choices (R_1, R_2) are made to penalize the input to guarantee that the output does not blow up and the system remains stable. Further details about the working of BO and its major components will be discussed in the next chapter.

Chapter 3

Hyperparameter Tuning using Bayesian Optimization

3-1 Bayesian Optimization

In this section, the general working of Bayesian Optimization (BO) and its components will be explored. The motivation for choosing certain methods is also discussed. In the end, the limitations of this approach are highlighted.

3-1-1 Block diagram of Bayesian optimization

The overview working of BO is represented in a block diagram in Figure 3-1. Let $c(x)$ be a function that needs to be minimised on some bounded set $\mathcal{X}$. Next is to construct a probabilistic model called the Surrogate Function (SF) for $c(x)$, then utilize this model to evaluate $c(x)$ by making decisions regarding the available best location inside $\mathcal{X}$. This process simultaneously takes into account the uncertainties and incorporates them. The model shapes the objective function in each iteration based on the points computed by the acquisition function. The next point is selected based on the distribution over the objective function and the trade-off between exploration (find new point) and exploitation (focus on the possible best points). The loop continues when the termination criteria are fulfilled or the number of iteration sets is exhausted.

The working of BO is given in the sequential steps below:

1. Setup the data points, initialization process and objective function.
2. Select the surrogate function. This function tailors an objective based on the initial points and returns an initial observation of this. This model is then updated iteration wise with new points.

3. The acquisition function selects the next possible best point. It is also responsible for the exploration and exploitation, in focus to find the points that provide an improved objective function.

4. The objective function is then evaluated based on the possible best points, and a new observation set of data points is generated that is given to the surrogate function in the next iteration

5. On selecting the next possible best points, the termination criteria are checked and the process is terminated if the criteria are met. Then the best points are fed to the evaluation process (MPC for this thesis) to generate the optimal results. Else, the algorithm proceeds to the next iteration.

3-1-2 Surrogate Function

A surrogate function is a model with a mathematical representation that approximates the unknown objective function. The surrogate function enables analysing the most favourable solution by offering a probabilistic estimation of the objective function in unobserved areas. The surrogate function is iteratively modified using the outcomes of each assessment of the objective function. Subsequently, the acquisition function is employed to determine the subsequent point for evaluation, considering the posterior distribution across the objective function. The main objective of surrogate functions is to reduce the number of expensive evaluations of the objective function required to determine the ideal solution.

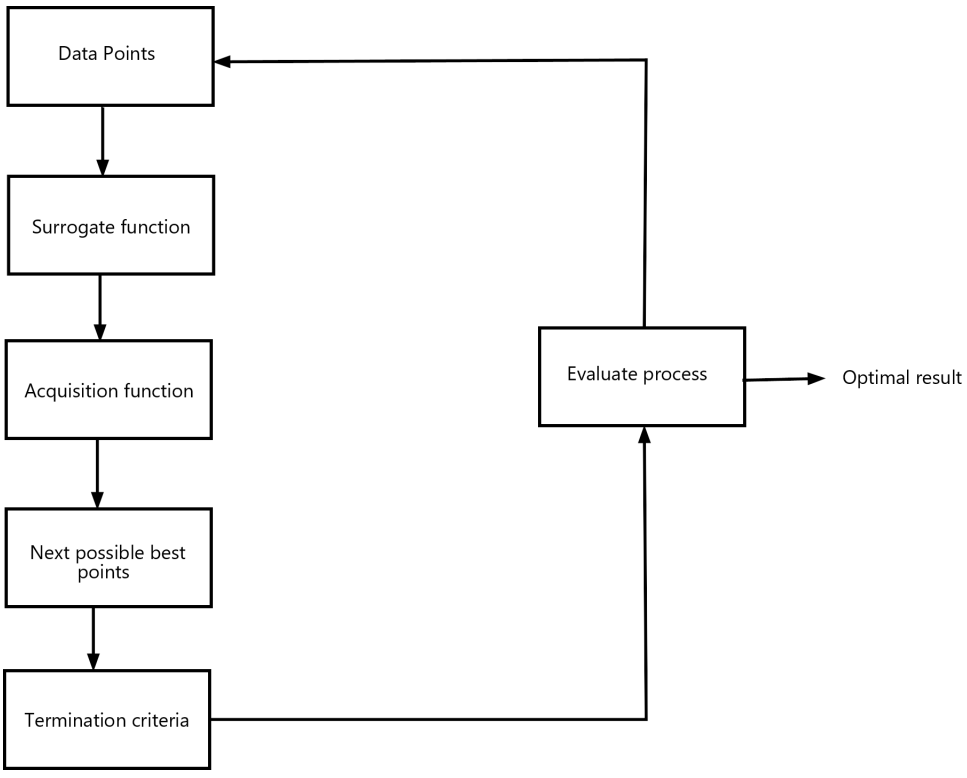


Figure 3-1: The block diagram of a generic Bayesian optimization approach

Random Forest (RF) (RF) and Gaussian Process (GP) are the two commonly used SFs [18]. There are several types of SFs apart from these two and another popular choice is the Bayesian Neural Network (BNN) explained in [71]. BNN combines Bayesian methods and neural network that provides a probabilistic framework for modelling uncertainty. Though BNN can provide flexibility and the ability to handle complexity, this method provides uncertainty estimates.

Compared to RF, GP is more suited for models with smooth objective functions and provides a probabilistic prediction of the objective function, allowing uncertainty quantification. GP establish a prior distribution over the objective function, which can be used to incorporate prior information towards the objective function. The evaluation of the GP is cost-effective compared to the other methods and is utilized in sampling new points, which results in better results however, has limitation when it comes to large data sets [18], [70]. Thus GP is the choice of surrogate function that is used in this thesis.

A GP starts by specifying a prior distribution over all functions that could potentially represent the objective function, without any initial assumptions based on specific knowledge of what form this function might take. This prior distribution is realized as the GP Prior over functions, and can entail an infinite number of decisions. As the algorithm sees more data points, this prior is updated to get a newer belief on the true objective function based off of our observed values. Under this paradigm, each input point is considered to itself be a random variable and the joint distribution of these variables fit with multivariate normal distributions. This in turn provides a flexible view of how the objective function is being modelled and updated, as seen by GPs. To guarantee the smoothness of the function kernel or covariance function is introduced and expressed as [27]:

$$k\left(x, x'\right) = \sigma^2 \exp\left(-\frac{1}{2\ell^2}\left\|x - x'\right\|^2\right) \tag{3-1}$$

where, x and x' are the two different point at which GP is being evaluated, ℓ is the horizontal

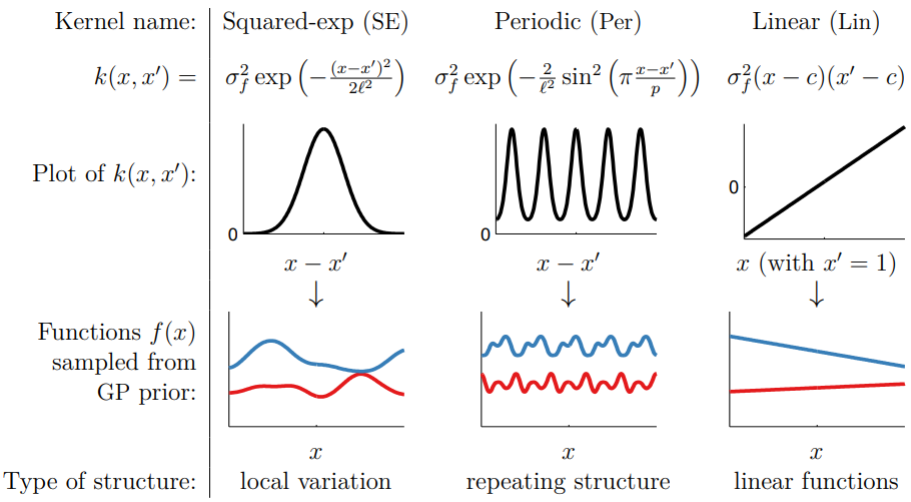


Figure 3-2: Example structure of a few kernels [27]

range and σ is the variance. In Figure 3-2, a few examples of kernels are shown. Once the kernel is computed, the covariance matrix $\Sigma(x, x')$ is constructed with a structure shown below:

$$\Sigma(x, x') = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) & \dots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) & \dots & k(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & k(x_N, x_3) & \dots & k(x_N, x_N) \end{bmatrix}$$

The covariance matrix must be positive definite. The sampling data y can be predicted from the following formula:

$$Y(x) \sim \mathcal{GP}(\mu, k(x, x')) \tag{3-2}$$

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}, \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

where, μ is the mean, Y_1 is the values to be calculated of y, Y_2 is the values from the training set, Σ_{11} is the covariance of test set and Σ_{22} is the covariance of training set. The distribution of Y_1 is conditional on $Y_2 = y_2$ is $Y_1 \mid y_2 \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma})$ given:

$$\bar{\mu} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y_2 - \mu_2)$$
$$\bar{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

With advantages, there are also a few limitations of GP. When the data points are large (a few thousand), the computation of the inverse of the covariance matrix is high, thus increasing the time complexity and slow inference. There is a degree of arbitrariness involved in the selection of a kernel. Non-Gaussian processes may be better appropriate for comprehending the events displayed by actual physical systems, as GP functions tend to generate perturbed starting points [27].

3-1-3 Acquisition function

The objective function is passed to the acquisition function after a posterior GP. The acquisition function then returns a real value which quantifies the expected utility of evaluating some point in the search space, and acts as a control variable that drives how to pick next point to evaluate. This acquisition function's main goal is to balance these two competing objectives of exploration and exploitation. This means choosing between exploitative promising points, or informative point for the learning signal of the optimisation objective [18].

Several types of acquisition function are mentioned in the literature survey, each having pros and cons. The most commonly known ones are Expected Improvement (EI), Probability of

Improvement (PI), Upper Confidence Bound (UCB), Lower Confidence Bound (LCB). In recent years, more research has come up with new acquisition functions with the motivation to improve the performance of BO, like Entropy Search (ES) [39], Predictive Entropy Search (PES) [40], and Max-Value Entropy Search (MES) [81]. However, these acquisition functions heavily rely on the sampling of GP and are computationally expensive. Hence, they are out of scope for this thesis work, and traditional acquisition functions such as EI, PI, UCB and LCB will be discussed further. The mathematical formulation and description of the acquisition functions are referred from paper [15].

Upper Confidence Bound

UCB is a simple acquisition function, and its mathematical formulation is given below:

$$UCB(x;\lambda) = \mu(x) + \lambda\sigma(x) \tag{3-3}$$

where, $\mu(x)$ and $\sigma(x)$ are the mean and the standard deviation of the GP posterior predictive at x for the function $f(x)$. They are also the exploitation and exploration terms, respectively, and the trade-off is regulated by tuning the parameter λ. Therefore, when λ is small, i.e. the $\mu(x)$ is high, the BO is expected to exploit the high-performing explored space. So, the UCB will be more conservative, resulting in aggressive sampling around the possible best points.

In the next case, when the λ is high, the $\sigma(x)$ is high, and the BO tends to explore more the uncharted areas in the search space. Here, the UCB will favour unexplored areas and passive around the already explored area. In the last scenario, when the value of λ is ≈ 1, the UCB is expected to perform a balanced approach between the known possible best values and unexplored areas in the search space. The LCB has a very similar approach. While the UCB is considered to be the optimistic approach. The LCB is considered to be the pessimistic approach because it is the subtraction of $\mu(x)$ and $\sigma(x)$.

Probability of Improvement

The objective function that needs to be maximized is $f(\mathbf{x})$, with $\mathbf{x}^+$ being the current best possible point. Then the PI can be defined as:

$$\mathrm{PI}(\mathbf{x}) = \max\left(f(\mathbf{x}) - f\left(\mathbf{x}^+\right), 0\right) \tag{3-4}$$

where, $f(\mathbf{x}^+)$ is the value of the current best solution. Now if the new $\mathbf{x}$ is such that the $f(\mathbf{x}) < f(\mathbf{x}^+)$, then the difference is negative however Equation 3-4 returns 0. This signifies that the BO is not improving. However, if $f(\mathbf{x}) > f(\mathbf{x}^+)$, then the difference is positive, and the BO will improve the current best solution by the difference. The PI can be evaluated analytically under the GP model:

$$\mathrm{PI}(\mathbf{x}) = \begin{cases} \phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}$$

where

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}$$

here, $\mu(\mathbf{x})$, $\sigma(\mathbf{x})$ and $\phi(Z)$ are the mean, standard deviation of the posterior GP at $\mathbf{x}$ and Cumulative Distribution Function (CDF) of the standard normal distribution respectively. An important observation that can be made here is that there is no explicit exploration function. This implies that PI primarily focus on exploitation and chance of improving the current value is uncertain.

Expected Improvement

The EI does not just consider the likelihood of improvement like PI but goes further to check the potential improvement. The mathematical definition of EI is given as follows:

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}\max(f(\mathbf{x}) - f\left(\mathbf{x}^{+}\right), 0\Big) \tag{3-5}$$

where $f(\mathbf{x}^{+})$ is the current best solution at x^{+} point which is defined as, $\mathbf{x}^{+} = \arg\max_{\mathbf{x}_i \in \mathbf{x}_{1:t}} f(\mathbf{x}_i)$. Here, $f(\mathbf{x}_i)$ is the value of the i^{th} iteration at x_i. Now, the analytical evaluation for EI is given as:

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} \left(\mu(\mathbf{x}) - f\left(\mathbf{x}^{+}\right) - \xi\right)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}$$

where

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^{+}\right) - \xi}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}$$

The formulation is similar to PI. However, EI has an exploration parameter ξ. In the equation above $\mu(\mathbf{x}) - f(\mathbf{x}^{+}) - \xi)\Phi(Z)$ part is the exploitation term and $\sigma(\mathbf{x})\phi(Z)$ part is the exploration term. The value of ξ determines the degree of exploration. Higher values will enable the BO to explore aggressively in the search space. Another significance of ξ is that as ξ increases, the significance of improvement projected by the posterior GP mean $\mu(x)$ diminishes in comparison to the significance of prospective enhancements in areas characterized by high prediction uncertainty, as shown by large $\sigma(x)$ values.

This leads to the conclusion of acquisition functions. UCB and EI can be explored further with the experiment setups in the thesis. As UCB and LCB are similar approaches, both need not be explored, hence choosing UCB (since it is an optimistic approach). Similarly, PI and EI are quite identical, and since EI can perform both exploration and exploitation, EI is chosen.

3-2 Optimization

The following section provides an introduction to BO and how it plays in the role of hyperparameter tuning. Finally, the last part of this section analyzes how to apply requirements and relates it back to a thesis.

3-2-1 Background on Bayesian Optimization

This section provides a preliminary overview of BO, with its primary significance lying in hyperparameter tuning. Several studies prove that BO performs better than other optimization algorithms like Predictive Entropy Search with Constraints (PESC), Constrained Optimization by Linear Approximation (COBYLA) in several benchmark functions, such as the Ackley function [31], [83], [30], [46]. Relative to gradient-based algorithms and brute-force search methods, BO has distinct advantages in the spaces where computing gradients is expensive or deals with optimization of multiple objective functions. It performs well in situations where functions are smooth but not necessarily convex. Unlike brute-force that searches exhaustively over a predefined set, BO leverages a probabilistic model to guide the search towards optimal solution. Then, since brute force search has a large spread in the space to find its optimal value it is computationally expensive and takes more time than BO. This is so because BO has a fundamentally efficient design that allows us to both explore efficiently while also exploiting effectively without as much concern for finding local optimal/suboptimal solutions [9].

3-2-2 Hyperparameter

Hyperparameters are the parameters which a user needs to set before the training of the model starts, unlike normal learning mechanisms in which your algorithm itself learns these weights (Among other weight type parameters). It controls various aspects of the behaviour of a learning algorithm, such as its learning rate; strength regularization and a number of hidden layers within a neural network. Choosing even the parameters of a neural network model has a traumatic impact on its performance and being able to find those most appropriate values becomes problematic. In hyperparameter tuning you systematically investigate a range, asking for the best performance on validation set or optimization problem minima. While one can also manually operate, this is work that we would usually want to automate and/or use multiple times [70].

Hyperparameters impact model performance greatly and choosing the right values for hyperparameter selection can be a challenging task. Hyperparameter tuning refers to a systematic exploration over a predetermined range of hyperparameter values, trying out their various combination until you find the best-performing one on some validation set. Selecting the best hyperparameters can increase the performance of this model and make it even more beneficial for doing its primary job. This has led to its various applications, especially in machine learning covering computer vision, image processing, speech recognition etc., and nowadays it is also serving as the preferred tool for hyperparameter tuning by a large number of researchers since BO has advantages over random or grid search are obvious. BO identifies optimal values while minimizing evaluations needed to learn algorithm [78].

3-2-3 Relevance to the Thesis

The utilization of BO has been extensively employed in the process of hyper-parameter tuning for deep learning, neural networks, and machine learning models. The same approach has been utilized to improve the performance of MPC [54], [55], [63], [78], [33]. In the context of

using MPC for soft robots, the motivation to opt for hyperparameter tuning using the BO approach for this thesis is the following:

1. Tuning performance parameters for an MPC via trial and error is time-consuming and exhaustive.
2. The performance of the MPC will improve in each iteration because of the evaluations of the closed-loop performance, hence limiting the model inaccuracy issues of the SR.
3. The functional relationship between the MPC and its tuning parameters is treated as a black box, and the proposed BO framework will determine the optimal parameter tuning by having a good trade-off between exploration and exploitation.
4. The utilization of this methodology can potentially improve the computational load over a simplistic grid search technique, hence offering a more effective means of optimizing MPC controller parameters.

Recognizing the benefits of hyperparameter tuning using the BO approach has led to the concept of automating the tuning process of the MPC's performance parameters.

3-2-4 Limitations

As BO comes with a lot of advantages, there are also a few limitations that need to be discussed. Some of the limitations are listed below:

1. When optimizing a high-dimensional objective function, BO gets computationally expensive. Utilization of complex surrogate function adds to the expense. Further, evaluating acquisition function and updating the surrogate function in every iteration needs for resources is drastically increased and is prohibitive as the objective function gets larger.
2. BO has limited scalability. As the number of variables and constraints increase, the search space grows exponentially. This makes the acquisition function less informative - which in turn impacts how well BO can do.
3. BO expects the objective function in such a way that it is smooth and continuous as most of real-world problem does not verify this hypothesis.
4. The definition of BO also requires defining different hyperparameters such as kernel function and its associated parameters, acquisition function with corresponding parameters. The optimal values of hyperparameters are a determining factor for the success BO, but they can be hard to specify.

While these constraints are present, BO has been widely used in many domains like engineering, finance and machine learning on a variety of theoretical optimization problems. Accordingly, BO may be an excellent choice to optimize complex and expensive functions if the surrogate model, acquisition function or hyperparameters are chosen carefully.

3-3 BO Implementation

This part of the thesis discusses the BO used to optimize the hyperparameters of an MPC. Figure 3-3 describes the block diagram of a closed-loop control system (MPC) using Bayesian Optimization. In this configuration, an MPC uses a mathematical model to forecast future states of the system and apply control inputs that would allow it to meet reference signals r. The strategy is well suited for scenarios when tuning the performance of the controller is cumbersome or takes a significant amount of time. The BO algorithm is tested using EI and LCB acquisition functions to understand which approach provides a better performance to the MPC. The method used to implement this algorithm is explained in the following steps:

- **Step 1:** Initialization

  The BO starts with the initialization of the hyperparameter. The weights for the MPC cost function Q and R are selected. The values are usually set with either having prior knowledge or sampled randomly within the range set which is 200 to 5000 for Q and 0.005 to 1 for R. For this experiment, the values chosen are $[Q_1, Q_4, R_1, R_2] = [300, 300, 1, 1]$ which is chosen through different iteration. The `ndgrid` function is used to create a 4D space defined by the range with linspace creating evenly spaced vectors for each parameter between the defined range. By defining this space the BO can evaluate the MPC performance at each point in the grid giving a broader understanding of the parameter space. This approach helps the BO algorithm to get the optimal region of the hyperparameter. The larger the space the better chance for the BO to not miss the optimal point however, this increases the computational expense. The next step is to define the objective function to evaluate how the MPC performs with different sets of weights which is evaluated in each iteration.

- **Step 2:** Surrogate model

  Using a Gaussian Process (GP) to model the relationship between MPC hyperparameters and observed performance metrics. The GP is fitted to data attained after repeatedly deploying the MPC initially at the start point and later on at the most attractive point according to the selected performance metric that is obtained from running the MPC in the previous iteration. The GP model provides mean performance prediction and standard deviation for the future hyperparameters.

- **Step 3:** Acquisition function optimization

  Expected Improvement (EI) works very well in finding the right trade-off between exploration and exploitation. Therefore could be one of the great choices for tuning an MPC. EI is calculated on the possible next points in hyperparameter space based upon performance till now. The predictor proceeds to choose points that will give, at expectation, the most improvement over best-observed performance taking mean prediction and variance as issued by the current GP model. For the next evaluation cycle, the parameters with the highest EI are taken.

  This exploration-exploitation trade-off is essentially the decision-making process in choosing whether it should explore into new areas with high uncertainty, or exploit what it already known from its experience to fine-tune some of the best solutions obtained so far. Once the MPC is tested with the new parameters, the GP model is

updated with new parameters to retain and update the GP model. This provides new predictions that are sampled around new points.

- **Step 4:** Iteration

  The process of predicting, optimizing, evaluating and updating continues. The MPC's performance tuning is further improved each iteration as the hyperparameter space better aligns with a global minimum. The optimization loop keeps running until a limit is met. This could be an all-around characterized number of cycles or computational limit. In this study, the iteration is set at 50. The set of parameters that perform best using the observed metrics are identified as optimal parameters for MPC. The optimum parameters are then fed to the MPC to attain the best control performance for the given conditions.

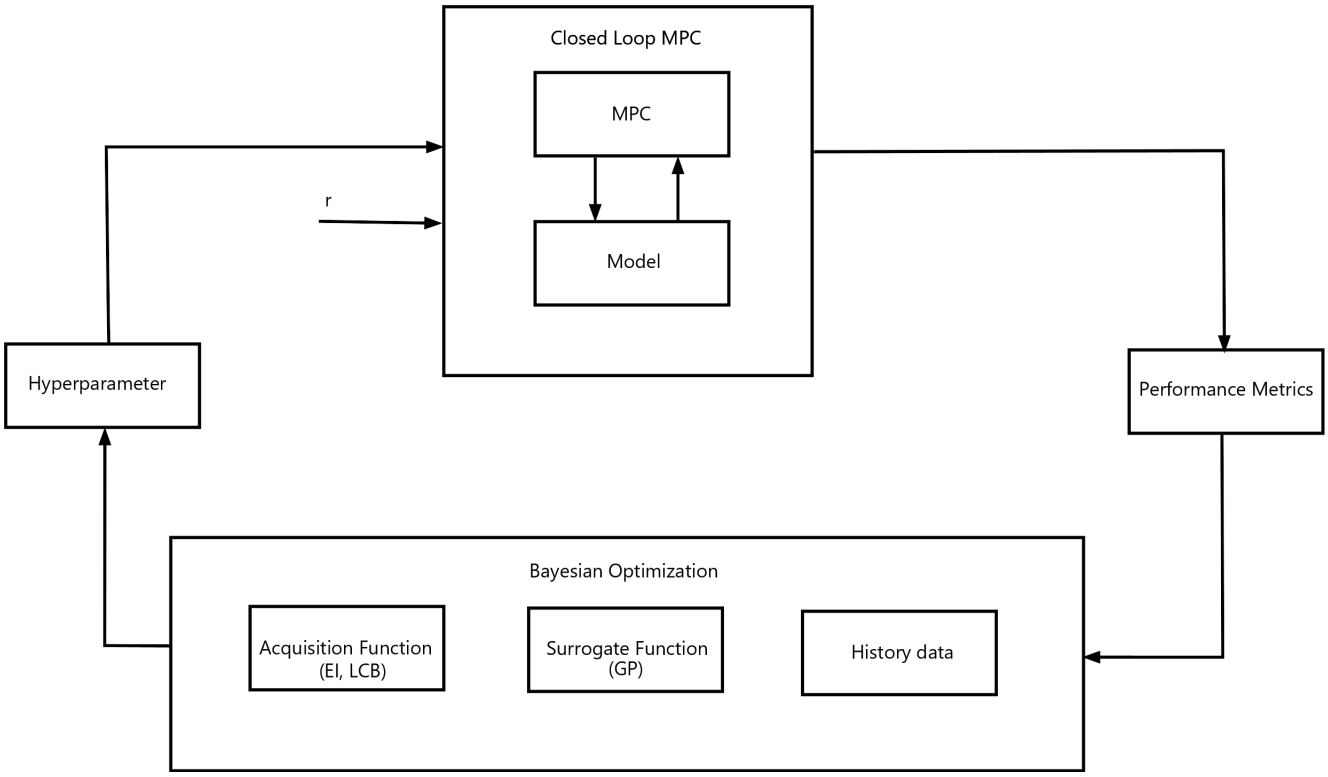


Figure 3-3: The block diagram of the implemented BO algorithm

Chapter 4

Simulations and Experiment Results

4-1 Experiment results

The chapter details the outcomes of simulations which are undertaken specifically to determine whether Bayesian optimization can be effectively employed for MPC parameter tuning in soft robotic systems. The emphasis is on tuning MPC parameters in a simulation environment to improve the control performance. The purpose of these simulations is to eliminate the sort trial-and-error manual tuning process that often results in sub-optimal control settings. The experiments conducted are for exploration-exploitation trade-off, robustness to initial conditions and BO under uncertainty. These experiments aim to provide a clear analysis of convergence trends, solution patterns, robustness to uncertainty and any loss in the quality of the solution.

4-1-1 Exploration-Exploitation

For the MPC setting the simulation time $N_{sim} = 30\,\text{s}$ and prediction horizon $N = 20$ is chosen as these values are crucial for the real-time control application. The performance of the MPC is evaluated based on several metrics, including tracking error, control effort, and computational experiments. These metrics are crucial for assessing the practical viability of the controller in real-world scenarios, even though the current study is limited to simulations. Table 4-1 summarize the results of the exploration-exploitation experiment for EI comparing the performance for three settings ($\zeta = 0.008$, $\zeta = 0.04$, $\zeta = 0.08$). The fast exploration setting ($\zeta = 0.08$), shows fastest convergence however, has the highest cost compared the rest. Figure 4-1 shows the convergence plot for all the 3 configurations. $\zeta = 0.008$ shown in Figure 4-1(a), this shows evidence for a high exploit focus, allowing the algorithm to refine solutions within one small area. The relatively flat line after the initial drop also indicates that not much exploration is done off of what was originally identified as good, also it finds the best solution late. As observed in figure(b), with $\zeta = 0.04$, the cost smoothly decreases and settles around minimum faster when compared to low exploration setting $\zeta = 0.008$. This results in a good trade-off between exploration and exploitation which allows for improved

ζ	**Cost**	**Convergence**	Q_1	Q_4	R_1	R_2
0.008	0.0072	48	1496	5000	0.005	0.005
0.04	0.0072	13	1592	5000	0.005	0.005
0.08	0.0073	11	200	2888	0.005	0.005

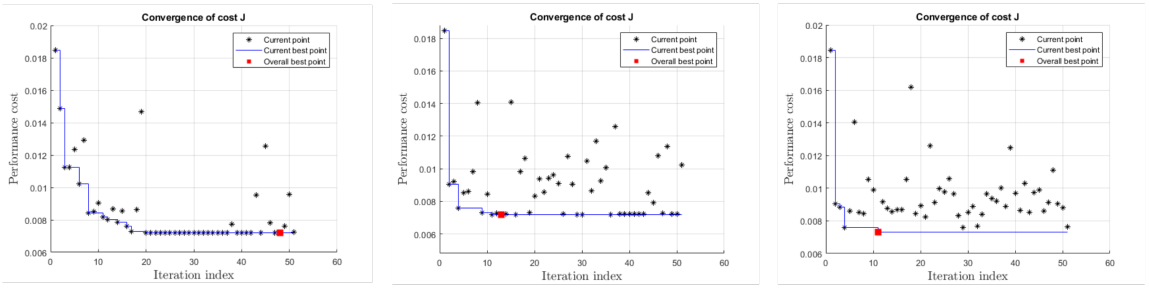
Table 4-1: Results of the exploration-exploitation experiment for EI

performance more consistently. Performance cost varies less in the long run, leading to being able to find and refine good enough solutions. In Figure 4-1(c) display a result of $\zeta = 0.08$. The cost drops sharply in initial exploration as shown here. It converges relatively faster than medium exploration $\zeta = 0.04$ with a higher cost. It can be observed that after getting the local best value the algorithm explores more missing the best optimum value.

The scatter plots Figure 4-2 illustrate the distribution and performance of different hyperparameter combinations Q_1 and Q_4. The medium exploration setting Figure 4-2(b) $\zeta = 0.04$, shows clusters of points achieving lower performance costs in the plots, indicating that reduced exploration still captures effective parameter regions efficiently. The very low exploration ($\zeta = 0.008$) however, seems to be too conservative, leading to slow convergence. Both low Figure 4-2(a) and high Figure 4-2(c) exploration settings seem to miss these optimal regions, either through insufficient exploration or by overextending into less effective parameter zones.

The cost values are very close across the experiments, suggesting that all three exploration settings are capable of finding effective hyperparameter configurations for the MPC. The lowest exploration setting ($\zeta = 0.008$) matches the medium setting ($\zeta = 0.04$) in achieving the lowest cost as shown in Figure 4-3, although it takes significantly more iterations to do so. The medium exploration setting ($\zeta = 0.04$) achieves the quickest convergence. This suggests that a moderate level of exploration can effectively balance the exploration-exploitation trade-off, quickly finding a near-optimal solution.

For the same experiment conducted for LCB using different values of $\lambda = 1$, $\lambda = 4.5$ and

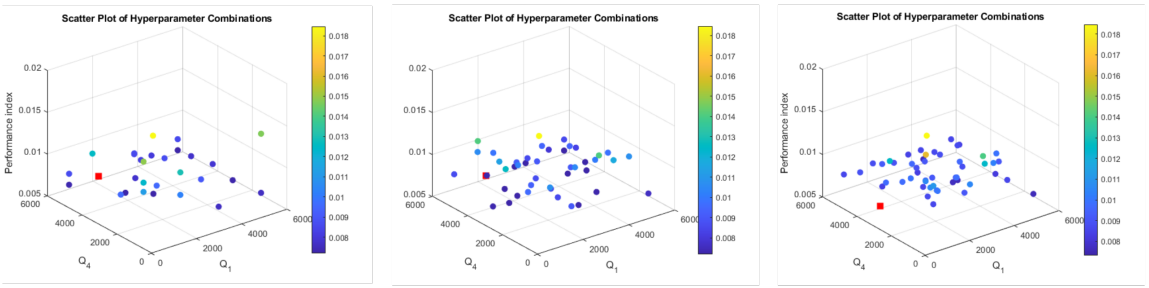


(a) (b) (c)

Figure 4-1: The convergence result of the exploration-exploitation experiment for EI. In the image (a)$\zeta = 0.008$, (b)$\zeta = 0.04$, (c)$\zeta = 0.08$

$\lambda = 10$, the results are shown in Table 4-2. The fastes convergence is shown for $\lambda = 1$, while the slowest is shown by $\lambda = 4.5$ however, with a lower cost. Meanwhile, $\lambda = 10$ strikes the middle ground both in terms of cost and convergence. Figure 4-4(a) shows rapid decrease to the lowest cost and maintains minimal fluctuations, suggesting that lower exploration focuses on quickly exploiting the promising areas found early in the search. Figure 4-4(b) shows for $\lambda = 1$, the slowest convergence but with a better performance cost than the $\lambda = 1$. There is a trade-off between the exploration-exploitation which provides a good result. For $\lambda = 10$ there is a quick drop in cost for Figure 4-4(c) and it helps in finding the best value at low cost indicating effective exploration of the parameter space.

Figure 4-5 shows the scatter plot that illustrates the distribution of Q_1 and Q_4 with the performance cost. Figure 4-5(a) shows the trend for exploitation which causes missing out on better-performing areas and achieving faster convergence. Therefore, focusing on exploiting



(a) (b) (c)

Figure 4-2: The scatter plot of the exploration-exploitation experiment for EI with (a)$\zeta = 0.008$, (b)$\zeta = 0.04$, (c)$\zeta = 0.08$

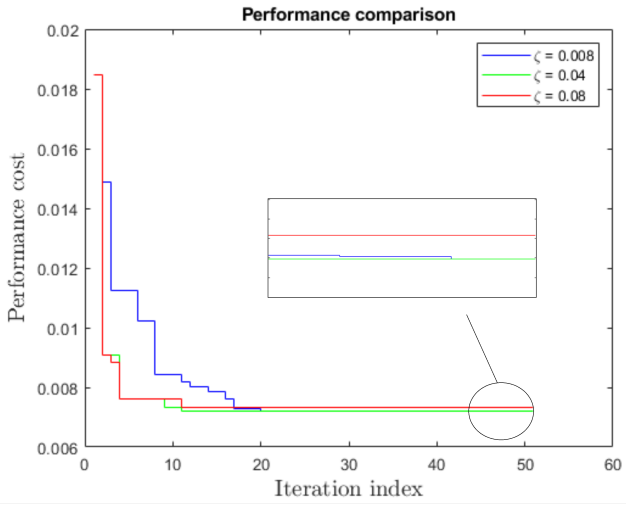


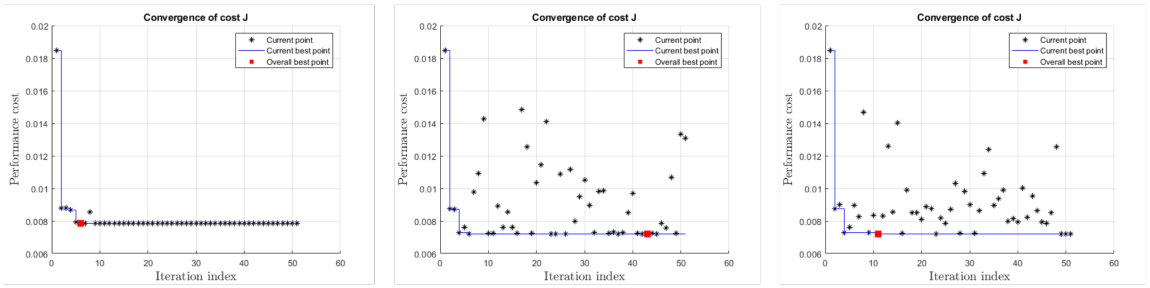
Figure 4-3: Performance comparison for different values of ζ. The magnified inset shows the highlight of the performance

λ	**Cost**	**Convergence**	Q_1	Q_4	R_1	R_2
1	0.0079	6	5000	5000	1	0.005
4.5	0.0072	43	968	3800	0.005	0.005
10	0.0072	11	632	4952	0.005	0.005

Table 4-2: Results of the exploration-exploitation experiment for LCB

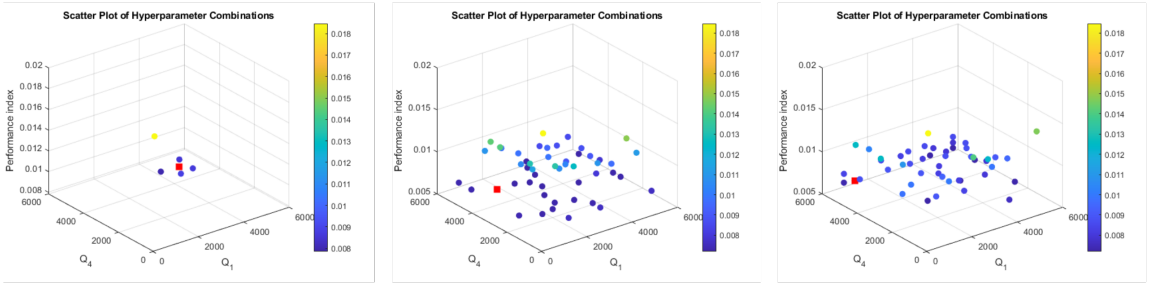
the best-known areas, lead to the quickest convergence. Figure 4-5(a) is a balanced approach of exploration with $\lambda = 4.5$. It has a slower convergence, as the algorithm spends more iterations evaluating a wider range of hyperparameters. Figure 4-6 shows the performance comparison where is it evident that the $\lambda = 4.5$ achieves the best cost by a small margin. Making it more evident that the BO performance is the best for this value of λ.

The result of the exploration-exploitation experiments shows that the different values of the



(a) (b) (c)

Figure 4-4: The convergence result of the exploration-exploitation experiment for EI. In the image (a)$\lambda = 1$, (b)$\lambda = 4.5$, (c)$\lambda = 10$



(a) (b) (c)

Figure 4-5: The scatter plot of the exploration-exploitation experiment for LCB with (a)$\lambda = 1$, (b)$\lambda = 4.5$, (c)$\lambda = 10$

exploration parameter have an influence on the performance cost and rate at which it converges to the best value. This is true for both the case of EI and LCB. With respect to the EI, larger ζ values resulted in wider exploration with very quick drop at the beginning, however especially for lower losses it was over-exploring (convergence less smooth). On the other hand, a medium ζ yielded the tradeoff of best solution quality: efficiently reaching optimal solution while managing costs effectively. Contrarily, a low ζ, which guarantees faster convergence at the cost of exploring broader regions in parameter space, indicates that we are most likely not sampling wide enough and hence I will be missing out better solutions.

In the case of LCB, the high λ value provided an extensive exploration, quickly identifying lower-cost regions but also leading to variability in performance, which might reflect an inefficient search pattern. Medium λ, while providing a comprehensive exploration, showed slower convergence. Low λ achieved the most rapid convergence and exhibited less change in cost, pointing to the exploitation of known good regions but with the inherent risk of settling into local optima.

While LCB with its high λ allowed for a wide exploration of the search space that identified low-performance cost regions rapidly, it left good performance. While Medium λ showed slower convergence but had a balanced exploration-exploitation. Convergence was fastest and cost had the least variation, suggesting it was more likely low λs just found a stable region in which to settle rather than actually exploring widely, even though it showed a good result, this need not be inconsistent with wide exploration then convergence on success.

From a high level, both EI and LCB are effective at the exploration-exploitation tradeoff, albeit to different degrees due to their differing parameter sensitivities. EI reacts to changes in ζ as it's the convergence behaviour that is mostly different across them, and for LCB such a change may be mitigated by adjusting λ, which will likely affect explored hyperparameters' distribution far greater than what an outlier can do with EI. Hence, the decision of whether to use EI or LCB in practice may depend on the properties of a given optimization: while for smoother landscapes and setting ζ properly with balanced exploration capability, then one

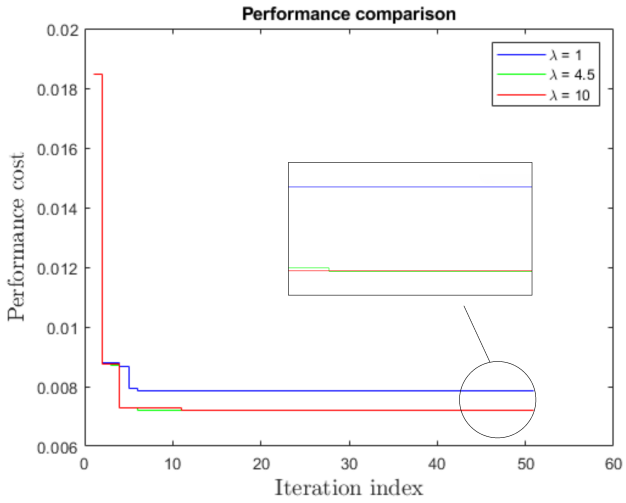


Figure 4-6: Performance comparison for different values of λ. The magnified inset shows the highlight of the performance

Condition	Initial point	Cost	Convergence
High Extremes	5000, 5000, 1, 1	0.0072	43
Low Extremes	100, 100, 0.001, 0.001	0.0072	13
Midpoint	2550, 2550, 0.5005, 0.5005	0.0072	28
Balanced 1	500, 5000, 0.1, 1	0.0072	41
Balanced 2	5000, 500, 1, 0.1	0.0072	27
Random point	300, 300, 1, 1	0.0072	13

Table 4-3: Results of the initial points experiment for EI

might prefer using EI; otherwise choosing LCB is advantageous especially if local minima are abundant.

4-1-2 Initial points

The BO is started with different values of initial points to evaluate the performance of the BO algorithm to the initial condition. The experiment aims to check the consistency in finding the optimum and convergence trend for different initial points. The starting points are selected from the possible range of parameter values. Table 4-1 shows the different conditions of the starting point selected and its corresponding result. The initial point is represented in the format $[Q_{sp1}, Q_{sp4}, R_{sp1}, R_{sp2}]$ where, Q_{sp1}, Q_{sp4}, R_{sp1}, R_{sp2} represents the initial points of parameters Q_1, Q_4, R_1 and R_2. Starting point has Extremes (high, low), Midpoint, Random Point and Balanced Points each of these to test how well BO algorithms can handle boundary conditions. The convergence for each condition is displayed in Figure 4-7 and Figure 4-8. Extreme high sees an initial spike downward in performance cost, which levels out. It also needed the longest training time among all runs, indicating that it was harder to now further refine the solution after these initial gains. Costs normally converge towards a single value quickly at the low extreme, because costs tend to stabilize and converges to the best value in very few iterations. Midpoint is one of the average convergence paths seeking to explore initial to check for all possible best values and exploits more after finding the region where it found the best point. This suggests a trade-off between the extremes, enabling to conduct an in depth exploration and exploitation. Balanced 1 convergence was slower than low extreme, and midpoint. The initial guess is that it relates to the wide scale of parameters making things complex to navigate. Balanced 2 converged faster than Balanced 1 which suggest that the composition of parameter scales in Balanced 2 was more optimal for optimization. Converged very quickly to the low cost, matching behavior that is seen in Low Extremes for Random point. Every single one of them eventually got down to roughly the same cost, indicating that EI is pretty resilient at finding a good solution no matter what you start with. Such a rugged conformity capacity is essential for practical use cases, where there are initial conditions that may not be ideal. Initially conditions also have a visible effect on how quickly the optimization is converging.

Table 4-4 shows the result of conducting the inital poitn experiment on LCB. Figure 4-9 and Figure 4-10 shows the convergence for different conditions. Low extreme provide efficiency

Condition	Initial point	Cost	Convergence
High Extremes	5000, 5000, 1, 1	0.0072	26
Low Extremes	100, 100, 0.001, 0.001	0.0073	1
Midpoint	2550, 2550, 0.5005, 0.5005	0.0086	4
Balanced 1	500, 5000, 0.1, 1	0.0072	5
Balanced 2	5000, 500, 1, 0.1	0.0072	21
Random point	300,300, 1, 1	0.0072	43

Table 4-4: Results of the initial points experiment for LCB

in finding the optimal cost however the later reading suggest that this initial point maybe not be reliable. Even for midpoint the convergence is fast however, the exploration for new points is not seen. High Extremes started with a sharp drop in performance cost and it can be noticed that the algorithm is has a better balanced approach compared to low extreme and midpoint. The conservative exploitation trend is seen even in both the balanced 1 and balanced 2 scenario with fast convergence. Though random point has a slower convergence, there is a well balanced exploration-exploitation making it more reliable.

Comparing these results with the EI method, several distinctions emerge. Base on the observation it is evident that EI provide more robustness by providing consistent performance cost and provides balanced exploration-exploitation. Meanwhile, LCB tends to converge more rapidly, particularly from lower or midpoint initial conditions. This makes LCB particularly effective where initial parameters are known to be sub-optimal or varied.

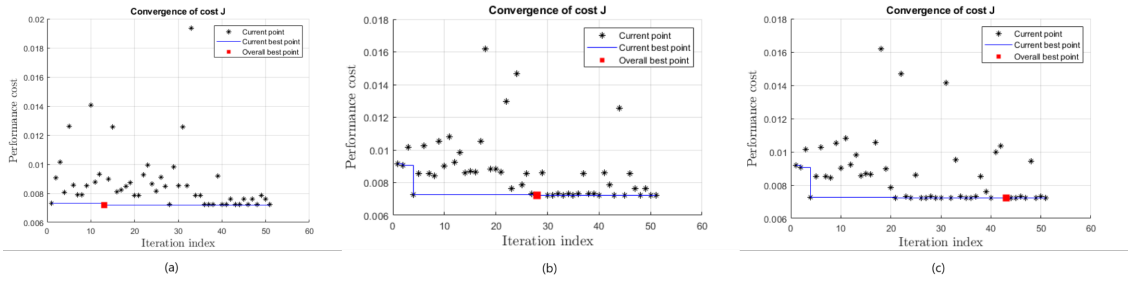


Figure 4-7: Initial point experiment for EI with (a)Low extreme, (b)Midpoint, (c)High extreme

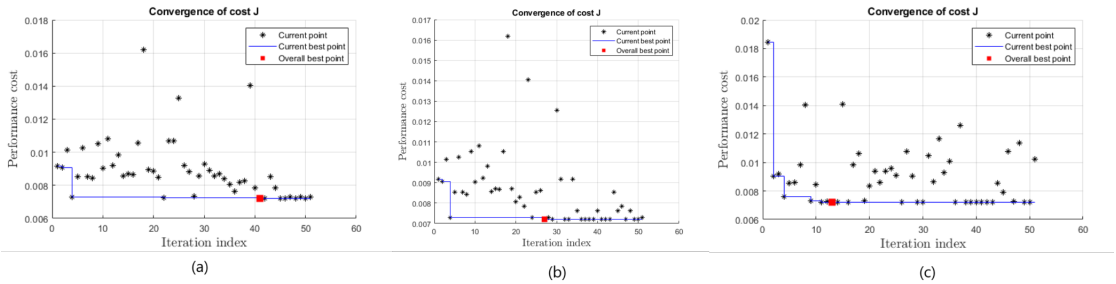


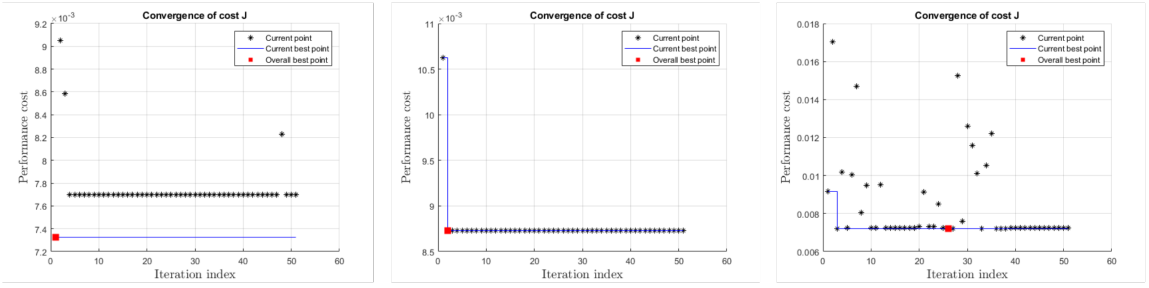
Figure 4-8: Initial point experiment for EI with (a)Balanced 1, (b)Balanced 2, (c)Random point

4-1-3 BO under uncertainty

The work in [68] introduces uncertainty to the MPC by introducing a variance (Σ_{un}) to the performance parameter Q. This formulation is given as:

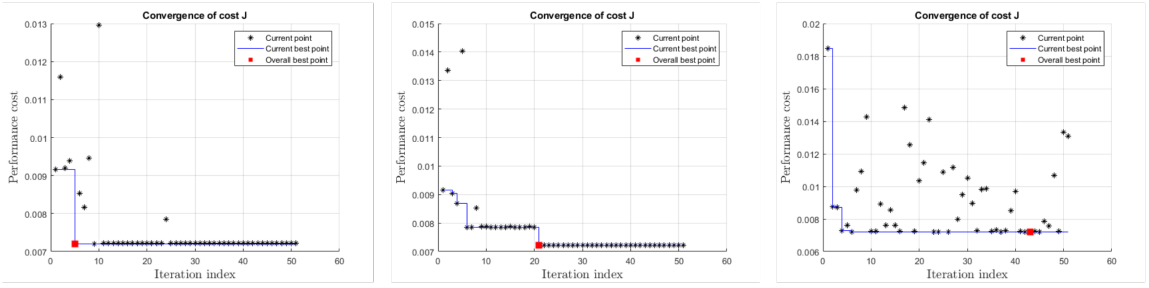
$$\begin{bmatrix} Q_{\alpha}^{*} \\ Q_{\beta}^{*} \end{bmatrix} \leftarrow \mathbf{Q_x}\,\mathrm{diag}^{-1}\left(\boldsymbol{\Sigma_{un}}\right) \tag{4-1}$$

In the context of this study, the Σ_{un} is integrated into the BO framework by applying $\Sigma_{un} = 0.05$ to the performance parameter of the MPC. This process is done in order to understand how BO would adapt to the uncertainty and analyse the performance. This specific Σ_{un} value is chosen to simulate moderate but significant uncertainty and reflect a scenario where parameters are not always perfectly known. This experiment is conducted on both EI and LCB. The initial point taken for this experiment is $[Q_{sp1}; Q_{sp4}, R_{sp1}, R_{sp2}] = [300; 300, 1, 1]$ with $\zeta = 0.04$ for EI and $\lambda = 4.5$. Figure 4-11b shows LCB has a drop in performance cost at the very early stage and quickly finds the best value which is a slightly higher cost of



(a) (b) (c)

Figure 4-9: Initial point experiment for LCB with (a)Low extreme, (b)Midpoint, (c)High extreme

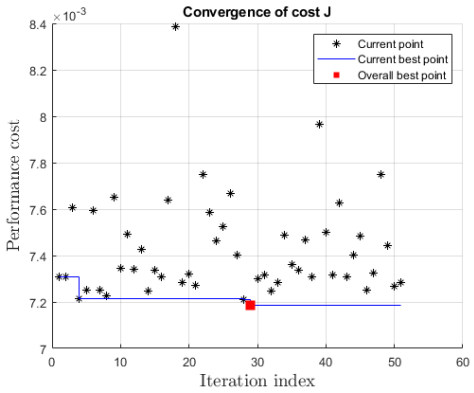


(a) (b) (c)

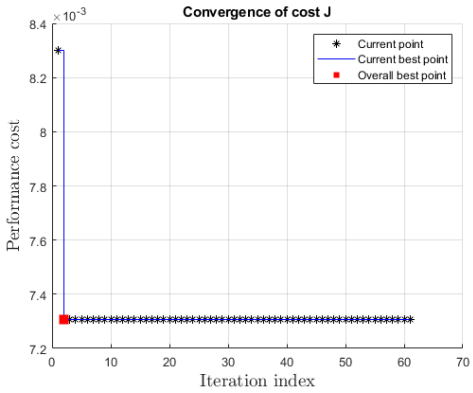
Figure 4-10: Initial point experiment for LCB with (a)Balanced 1, (b)Balanced 2, (c)Random point

0.0073 compared to EI and achieving convergence in just 2 iterations. This fast convergence suggests that the LCB method quickly locates near-optimal solutions even under the presence of significant uncertainty for the given setup. The results shown in Figure 4-11a are for the EI method which although at a slower rate, achieves convergence in 29 iterations with a slightly lower final cost of 0.0072. This shows that while EI explores more broadly, it may provide marginally better performance at the expense of finding the best value at a slower rate. LCB has an aggressive approach to finding the best value is likely due to its nature of exploitation with cautious exploration however could be stuck at local minima. In contrast, EI's gradual convergence showcases its balanced approach to exploring a large range of solutions, potentially providing a better understanding of the parameter space before settling on the optimal value.

Analysis of EI and LCB in comparison with the analysis performed under uncertainty (i.e. variance in MPC performance matrix) demonstrates strengths as well as appropriateness based on individual operational requirements. Due to LCB's quick adaptation, it is recommended for applications of fast response as it converges to the best value quickly. In contrast, EI's method is better suited to applications where achieving the best performance cost is a priority as it gives the best performance cost compared to LCB however, slow at finding the best value.



(a) The plot represent the result for EI for the BO under uncertainty experiment



(b) The plot represent the result for LCB for the BO under uncertainty experiment

Figure 4-11: The plot represent the result for EI (a) and LCB (b) for the BO under uncertainty experiment

Chapter 5

Conclusion

This chapter closes with the final remarks on our exploration of Bayesian Optimization (BO) for Model Predictive Control (MPC) systems in soft robotics. It provides a brief synopsis of what this research has contributed, linking the concrete cases studied in Chapter 4 back to the motivation stated in Chapter 1. This synthesis is intended to assess the extent of which the proposed practice solves those initial problems highlighted at step one for being addressed in this study. Moreover, this chapter will elaborate a set of prospective directions for future research to put forward how additional works can make use of our contributions in order to improve and generalize the parameter tuning possibilities within BO.

5-1 Summary and Conclusion

The thesis investigates using BO to tune the hyperparameters of an MPC for soft robots. However, these soft robots offer the challenges of non-linear highly compliant structures over traditional control strategies employed for rigid designs. The reason for using BO are that the tuning of MPC, which is usually done manually (which takes time and can lead to sub-optimal performance) It should be automated. The complexity associated with high degrees of freedom and nonlinear material properties makes the control more challenging for soft robots, requiring an adaptive strategy able to respond dynamically when conditions change without needing extensive manual tuning.

Performance parameters of the MPC, specifically its cost functions within control strategy are optimized using BO. The approach is based on an iterative algorithm in which a Gaussian Process (GP) surrogate model estimates performance metrics for different hyperparameter configurations. These acquisition functions, especially Expected Improvement (EI) and Lower Confidence Bound (LCB), help in choosing the next parameter sets to evaluate trying to find a compromise between exploring new areas of parameters space against exploiting know good ones. Several experiments conducted to evaluate the effectiveness of BO in different settings are as follows:

- Exploration-Exploitation: Experiments were conducted for different values of exploration. Satisfactory results were found for the balanced approach, for EI $\zeta = 0.04$ and for LCB $\lambda = 4.5$. These values yield the most consistent performance improvements across iterations.
- Robustness to Initial Conditions: Performance of the algorithm was tested over various initial points served as validation checkpoints. EI was consistent with optimal cost but showed slow convergence. While LCB showed fast convergence however, the performance cost was comparatively higher.
- BO Under Uncertainty: To model real-world uncertainties, we introduced variance in performance parameters and found that both EI and LCB could adapt well to the changing scenarios efficiently; while LCB converged faster than EI, but with a higher cost of optimality.

To sum up, BO enabled a great step forward for soft robotics research. It presents a new methodological gain for soft robots and could provide them with the ability to operate in numerous fields, including both medical and industrial contexts. Lending insights into autonomous tuning methods, the research also inspires further investigation about what these results mean for how we might design and operate complex robotic systems.

5-2 Limitations and Future work

This section explores the limitations that the BO-based MPC faces along with a brief outline of future research to improve this method further. In the current context, the implementation is limited to linear models for the MPC. This is an informative way to derive the fundamental dynamics and helps simplify subsequent optimization but will not generalize properly establish all possible behaviours for more advanced soft robotic systems. This is primarily due to the computational demands and non-linearity such models often introduce into optimization, but even more importantly the linear approximations have been shown as inadequate in accurately representing real-world dynamics. BO is computationally expensive in general, and even more so with a GP that must be updated frequently. The current computation framework might not be able to scale well with more complicated models or the added hyperparameters. Such limitations are important in scenarios with constrained real-time computing capabilities.

Increasing the parameter space is required to make most of BO in MPC optimisation. Nonetheless, the applicability of BO to large parameter spaces is still quite limited as was documented in this thesis. This constraint highlights the pressing demand for algorithms of greater efficiency in order to handle higher-dimensional spaces while remaining computationally tractable. Disturbance rejection performance of the optimized MPC was not a primary design objective in this work. Rarely, in reality, a measure of a controllers worth is how well it can handle unknown perturbations. The extensions of these disturbances rejection abilities to the optimizations criteria may be advantageous in future studies, improving control strategies practical robustness.

This thesis mainly focused on utilizing EI and LCB as acquisition functions separately. However, there is new research wherein a new method is introduced which is a mixed approach

combining EI and PI [8]. This method claims to provide better handling of complex systems and faster convergence due to their balanced approach towards exploration. Second, considering alternative scalability enhancements for BO with more challenging nonlinear dynamics by potentially including computationally cheaper algorithms or combinations of optimization strategies in hybrid settings would be an interesting avenue as well. This was not only helpful in dealing with computational constraints, but would also lead to improvement of the robustness and adaptability of the systems being optimized by the MPC.

In summary, although the presented thesis builds a very good baseline for exploiting BO with MPC systems and has outlined limitations as well as future directions to provide opportunities towards more holistic studies. Ideally, these studies would be one step closer to the practical realization of BO by providing efficient and general control solutions for complex and dynamic robotic systems.

Appendix A

Appendix

A-1 Model characteristics of the soft robot

Poles of the model:

$$0.9989 + 0.0268i$$
$$0.9989 - 0.0268i$$
$$0.0013 + 0.0000i$$
$$0.9995 + 0.0155i$$
$$0.9995 - 0.0155i$$
$$0.0013 + 0.0000i$$

Poles of the Plant:

$$0.9989 + 0.0244i$$
$$0.9989 - 0.0244i$$
$$0.0048 + 0.0000i$$
$$0.9995 + 0.0145i$$
$$0.9995 - 0.0145i$$
$$0.0048 + 0.0000i$$

A-2 Simulation environment

The simulations were ran on a machine equipped with an Intel i7-7700HQ CPU @ 2.80GHz, with 16 GB of RAM, Windows 10, and Matlab R2022a. The MPC was implemented using CasADi v3.6.3.

Bibliography

[1] Yasmin Ansari, Mariangela Manti, Egidio Falotico, Matteo Cianchetti, and Cecilia Laschi. Multiobjective optimization for stiffness and position control in a soft robot arm module. *IEEE Robotics and Automation Letters*, 3(1):108–115, 2017.

[2] Oluwaseun A Araromi, Irina Gavrilovich, Jun Shintake, Samuel Rosset, Muriel Richard, Volker Gass, and Herbert R Shea. Rollable multisegment dielectric elastomer minimum energy structures for a deployable microsatellite gripper. *IEEE/ASME Transactions on mechatronics*, 20(1):438–446, 2014.

[3] Costanza Armanini, Frédéric Boyer, Anup Teejo Mathew, Christian Duriez, and Federico Renda. Soft robots modeling: A structured overview. *IEEE Transactions on Robotics*, 2023.

[4] T Ashuri, A Armani, R Jalilzadeh Hamidi, T Reasnor, S Ahmadi, and K Iqbal. Biomedical soft robots: Current status and perspective. *Biomedical Engineering Letters*, 10:369–385, 2020.

[5] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomlin. Goal-driven dynamics learning via bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5168–5173. IEEE, 2017.

[6] Guanjun Bao, Hui Fang, Lingfeng Chen, Yuehua Wan, Fang Xu, Qinghua Yang, and Libin Zhang. Soft robotics: Academic insights and perspectives through bibliometric analysis. *Soft robotics*, 5(3):229–241, 2018.

[7] Marc Behl and Andreas Lendlein. Shape-memory polymers. *Materials today*, 10(4):20–28, 2007.

[8] Carolin Benjamins, Elena Raponi, Anja Jankovic, Koen van der Blom, Maria Laura Santoni, Marius Lindauer, and Carola Doerr. Pi is back! switching acquisition functions in bayesian optimization. *arXiv preprint arXiv:2211.01455*, 2022.

[9] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[10] Charles M Best, Morgan T Gillespie, Phillip Hyatt, Levi Rupert, Vallan Sherrod, and Marc D Killpack. A new soft robot control method: Using model predictive control for a pneumatically actuated humanoid. *IEEE Robotics & Automation Magazine*, 23(3):75–84, 2016.

[11] Eivind Bøhn, Sebastien Gros, Signe Moe, and Tor Arne Johansen. Optimization of the model predictive control meta-parameters through reinforcement learning. *Engineering Applications of Artificial Intelligence*, 123:106211, 2023.

[12] Pablo Borja, Azita Dabiri, and Cosimo Della Santina. Energy-based shape regulation of soft robots with unactuated dynamics dominated by elasticity. In *2022 IEEE 5th international conference on soft robotics (RoboSoft)*, pages 396–402. IEEE, 2022.

[13] Alexander S Boxerbaum, Kendrick M Shaw, Hillel J Chiel, and Roger D Quinn. Continuous wave peristaltic motion in a robot. *The international journal of Robotics Research*, 31(3):302–318, 2012.

[14] Frédéric Boyer, Mathieu Porez, and Wisama Khalil. Macro-continuous computed torque algorithm for a three-dimensional eel-like robot. *IEEE transactions on robotics*, 22(4):763–775, 2006.

[15] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[16] Daniel Bruder, Brent Gillespie, C David Remy, and Ram Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. *arXiv preprint arXiv:1902.02827*, 2019.

[17] Daniel Bruder, C David Remy, and Ram Vasudevan. Nonlinear system identification of soft robot dynamics using koopman operator theory. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6244–6250. IEEE, 2019.

[18] Antonio Candelieri. A gentle introduction to bayesian optimization. In *2021 Winter Simulation Conference (WSC)*, pages 1–16. IEEE, 2021.

[19] Hao Cheng, Houde Liu, Xueqian Wang, and Bin Liang. Approximate piecewise constant curvature equivalent model and their application to continuum robot configuration estimation. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1929–1936. IEEE, 2020.

[20] Sheng Cheng, Minkyung Kim, Lin Song, Chengyu Yang, Yiquan Jin, Shenlong Wang, and Naira Hovakimyan. Difftune: Auto-tuning through auto-differentiation. *arXiv arXiv:2209.10021*, 2022.

[21] Cosimo Della Santina, Antonio Bicchi, and Daniela Rus. On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control. *IEEE Robotics and Automation Letters*, 5(2):1001–1008, 2020.

[22] Cosimo Della Santina, Antonio Bicchi, and Daniela Rus. On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control. *IEEE Robotics and Automation Letters*, 5(2):1001–1008, 2020.

[23] Cosimo Della Santina, Manuel G Catalano, Antonio Bicchi, M Ang, O Khatib, and B Siciliano. Soft robots. *Encyclopedia of Robotics*, 489, 2020.

[24] Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model-based control of soft robots: A survey of the state of the art and open challenges. *IEEE Control Systems Magazine*, 43(3):30–65, 2023.

[25] Cosimo Della Santina, Robert K Katzschmann, Antonio Bicchi, and Daniela Rus. Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment. *The International Journal of Robotics Research*, 39(4):490–513, 2020.

[26] Cosimo Della Santina, Robert K Katzschmann, Antonio Biechi, and Daniela Rus. Dynamic control of soft robots interacting with the environment. In *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pages 46–53. IEEE, 2018.

[27] David Duvenaud. *Automatic model construction with Gaussian processes*. Doctoral thesis, University of Toronto, 2014.

[28] William Edwards, Gao Tang, Giorgos Mamakoukas, Todd Murphey, and Kris Hauser. Automatic tuning for data-driven model predictive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7379–7385. IEEE, 2021.

[29] William Edwards, Gao Tang, Giorgos Mamakoukas, Todd Murphey, and Kris Hauser. Automatic tuning for data-driven model predictive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7379–7385, 2021.

[30] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[31] David Eriksson and Matthias Poloczek. Scalable constrained bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 730–738. PMLR, 2021.

[32] Raony M Fontes, Maércio AF Martins, and Darci Odloak. An automatic tuning method for model predictive control strategies. *Industrial & Engineering Chemistry Research*, 58(47):21602–21613, 2019.

[33] Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded mpc. *IFAC-PapersOnLine*, 53(2):5189–5194, 2020.

[34] Jennifer Frame, Nick Lopez, Oscar Curet, and Erik D Engeberg. Thrust force characterization of free-swimming soft robotic jellyfish. *Bioinspiration & biomimetics*, 13(6):064001, 2018.

[35] Thomas George Thuruthel, Yasmin Ansari, Egidio Falotico, and Cecilia Laschi. Control strategies for soft robotic manipulators: A survey. *Soft robotics*, 5(2):149–163, 2018.

[36] Seyede Fatemeh Ghoreishi, Ryan D Sochol, Dheeraj Gandhi, Axel Krieger, and Mark Fuge. Bayesian optimization for design of multi-actuator soft catheter robots. *IEEE transactions on medical robotics and bionics*, 3(3):725–737, 2021.

[37] Nathaniel N Goldberg, Xiaonan Huang, Carmel Majidi, Alyssa Novelia, Oliver M O'Reilly, Derek A Paley, and William L Scott. On planar discrete elastic rod models for the locomotion of soft robots. *Soft robotics*, 6(5):595–610, 2019.

[38] Malte Grube, Jan Christian Wieck, and Robert Seifried. Comparison of modern control methods for soft robots. *Sensors*, 22(23):9464, 2022.

[39] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[40] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

[41] Matthias Hofer and Raffaello D'Andrea. Design, fabrication, modeling and control of a fabric-based spherical robotic arm. *Mechatronics*, 68:102369, 2020.

[42] Yaohui Huang, Matthias Hofer, and Raffaello D'Andrea. Offset-free model predictive control: A ball catching application with a spherical soft robotic arm. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 563–570. IEEE, 2021.

[43] Phillip Hyatt, Curtis C Johnson, and Marc D Killpack. Model reference predictive adaptive control for large-scale soft robots. *Frontiers in Robotics and AI*, 7:558027, 2020.

[44] Alex S. Ira, Iman Shames, Chris Manzie, Robert Chin, Dragan Nešić, Hayato Nakada, and Takeshi Sano. A machine learning approach for tuning model predictive controllers. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 2003–2008, 2018.

[45] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets riemannian manifolds in robot learning. In *Conference on Robot Learning*, pages 233–246. PMLR, 2020.

[46] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets riemannian manifolds in robot learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 233–246. PMLR, 30 Oct–01 Nov 2020.

[47] Curtis C Johnson, Tyler Quackenbush, Taylor Sorensen, David Wingate, and Marc D Killpack. Using first principles for deep learning and model-based control of soft robots. *Frontiers in Robotics and AI*, 8:654398, 2021.

[48] Brian Byunghyun Kang, Daekyum Kim, Hyungmin Choi, Useok Jeong, Kyu Bum Kim, Sungho Jo, and Kyu-Jin Cho. Learning-based fingertip force estimation for soft wearable hand robot with tendon-sheath mechanism. *IEEE Robotics and Automation Letters*, 5(2):946–953, 2020.

[49] Ameer Hamza Khan and Shuai Li. Sliding mode control with pid sliding surface for active vibration damping of pneumatically actuated soft robots. *IEEE Access*, 8:88793–88800, 2020.

[50] Daekyum Kim, Sang-Hun Kim, Taekyoung Kim, Brian Byunghyun Kang, Minhyuk Lee, Wookeun Park, Subyeong Ku, DongWook Kim, Junghan Kwon, Hochang Lee, et al. Review of machine learning methods in soft robotics. *Plos one*, 16(2):e0246102, 2021.

[51] Cecilia Laschi, Matteo Cianchetti, Barbara Mazzolai, Laura Margheri, Maurizio Follador, and Paolo Dario. Soft robot arm inspired by the octopus. *Advanced robotics*, 26(7):709–727, 2012.

[52] Chiwon Lee, Myungjoon Kim, Yoon Jae Kim, Nhayoung Hong, Seungwan Ryu, H Jin Kim, and Sungwan Kim. Soft robot review. *International Journal of Control, Automation and Systems*, 15:3–15, 2017.

[53] Junn Yong Loo, Chee Pin Tan, and Surya Girinatha Nurzaman. H-infinity based extended kalman filter for state estimation in highly non-linear soft robotic system. In *2019 American control conference (ACC)*, pages 5154–5160. IEEE, 2019.

[54] Qiugang Lu, Ranjeet Kumar, and Victor M Zavala. Mpc controller tuning using bayesian optimization techniques. *arXiv preprint arXiv:2009.14175*, 2020.

[55] Alberto Lucchini, Simone Formentin, Matteo Corno, Dario Piga, and Sergio M Savaresi. Torque vectoring for high-performance electric vehicles: an efficient mpc calibration. *IEEE Control Systems Letters*, 4(3):725–730, 2020.

[56] Andrew D Marchese, Cagdas D Onal, and Daniela Rus. Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. *Soft robotics*, 1(1):75–87, 2014.

[57] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic lqr tuning based on gaussian process global optimization. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 270–277. IEEE, 2016.

[58] L Margheri, C Laschi, and B Mazzolai. Soft robotic arm inspired by the octopus: I. from biological functions to artificial requirements. *Bioinspiration & biomimetics*, 7(2):025004, 2012.

[59] Barbara Mazzolai, Laura Margheri, Matteo Cianchetti, Paolo Dario, and Cecilia Laschi. Soft-robotic arm inspired by the octopus: Ii. from artificial requirements to innovative technological solutions. *Bioinspiration & biomimetics*, 7(2):025005, 2012.

[60] Mostafa A Mousa, Mennaallah Soliman, Mahmood A Saleh, and Ahmed G Radwan. Biohybrid soft robots, e-skin, and bioimpedance potential to build up their applications: a review. *IEEE Access*, 8:184524–184539, 2020.

[61] W David Null, William Edwards, Dohun Jeong, Teodor Tchalakov, James Menezes, Kris Hauser, et al. Automatically-tuned model predictive control for an underwater soft robot. *IEEE Robotics and Automation Letters*, 2023.

[62] Anuj Pal, Tianyi He, and Wenpeng Wei. Sample-efficient model predictive control design of soft robotics by bayesian optimization. *arXiv preprint arXiv:2210.08780*, 2022.

[63] Dario Piga, Marco Forgione, Simone Formentin, and Alberto Bemporad. Performance-oriented model learning for data-driven mpc design. *IEEE control systems letters*, 3(3):577–582, 2019.

[64] Federico Renda, Vito Cacucciolo, Jorge Dias, and Lakmal Seneviratne. Discrete cosserat approach for soft robot dynamics: A new piece-wise constant strain model with torsion and shears. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5495–5502. IEEE, 2016.

[65] Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553):467–475, 2015.

[66] SM Hadi Sadati, Seyedeh Elnaz Naghibi, Ali Shiva, Ian D Walker, Kaspar Althoefer, and Thrishantha Nanayakkara. Mechanics of continuum manipulators, a comparative study of five methods with experiments. In *Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017, Guildford, UK, July 19–21, 2017, Proceedings 18*, pages 686–702. Springer, 2017.

[67] Joseé Eduardo W Santos, Jorge Otaévio Trierweiler, and Marcelo Farenzena. Model predictive control tuning strategy for non-square systems and range controlled variables based on multi-scenarios approach. *Industrial & Engineering Chemistry Research*, 56(40):11496–11506, 2017.

[68] Alessandro Saviolo, Jonathan Frey, Abhishek Rathod, Moritz Diehl, and Giuseppe Loianno. Active learning of discrete-time dynamics for uncertainty-aware model predictive control. *IEEE Transactions on Robotics*, 2023.

[69] Dale E Seborg, Thomas F Edgar, Duncan A Mellichamp, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 2016.

[70] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[71] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29, 2016.

[72] Koichi Suzumori, Satoshi Endo, Takefumi Kanda, Naomi Kato, and Hiroyoshi Suzuki. A bending pneumatic rubber actuator realizing soft-bodied manta swimming robot. In *Proceedings 2007 IEEE international conference on robotics and automation*, pages 4975–4980. IEEE, 2007.

[73] Chee Pin Tan, Surya Nurzaman, Junn Yong Loo, and Ze Yang Ding. Future trends in i&m: Indirect sensing in soft robots using observers/filters. *IEEE Instrumentation & Measurement Magazine*, 23(1):42–43, 2020.

[74] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[75] Maxime Thieffry, Alexandre Kruszewski, Christian Duriez, and Thierry-Marie Guerra. Control design for soft robots based on reduced-order model. *IEEE Robotics and Automation Letters*, 4(1):25–32, 2018.

[76] Deepak Trivedi, Christopher D Rahn, William M Kier, and Ian D Walker. Soft robotics: Biological inspiration, state of the art, and future research. *Applied bionics and biomechanics*, 5(3):99–117, 2008.

[77] Björn Verrelst, Ronald Van Ham, Bram Vanderborght, Frank Daerden, Dirk Lefeber, and Jimmy Vermeulen. The pneumatic biped "lucy" actuated with pleated pneumatic artificial muscles. *Autonomous Robots*, 18:201–213, 2005.

[78] A Helen Victoria and Ganesh Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12:217–223, 2021.

[79] Alex Villanueva, Colin Smith, and Shashank Priya. A biomimetic robotic jellyfish (robojelly) actuated by shape memory alloy composite actuators. *Bioinspiration & biomimetics*, 6(3):036004, 2011.

[80] Jue Wang and Alex Chortos. Control strategies for soft robot systems. *Advanced Intelligent Systems*, 4(5):2100165, 2022.

[81] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[82] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

[83] Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. *Advances in neural information processing systems*, 30, 2017.

[84] Andre Shigueo Yamashita, Antônio Carlos Zanin, and Darci Odloak. Tuning of model predictive control with multi-objective optimization. *Brazilian Journal of Chemical Engineering*, 33:333–346, 2016.

[85] Stanislaw H Zak. An introduction to model-based predictive control (mpc). *lecture notes on ECE680, Purdue University*, 2017.

[86] Zhongkai Zhang, Jeremie Dequidt, Alexandre Kruszewski, Frederick Largilliere, and Christian 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)*, pages 5509–5514. IEEE, 2016.

Glossary

List of Acronyms

MPC	Model Predictive Controller
BO	Bayesian Optimization
CC	Constant Curvature
PCC	Piecewise Constant Curvature
APCC	Approximate Piecewise Constant Curvature
FEM	Finite Element Method
POD	Proper Orthogonal Decomposition
ILC	Iterative Learning Control
EKF	Extended Kalman Filter
ZOH	Zero-order Hold
UB	Upper Bound
LB	Lower Bound
OCP	Optimal Control Problem
QP	Quadratic Programming
SF	Surrogate Function
RF	Random Forest (RF)
GP	Gaussian Process
BNN	Bayesian Neural Network
EI	Expected Improvement
PI	Probability of Improvement
UCB	Upper Confidence Bound
LCB	Lower Confidence Bound
PES	Predictive Entropy Search

ES	Entropy Search
MES	Max-Value Entropy Search
CDF	Cumulative Distribution Function
PESC	Predictive Entropy Search with Constraints
COBYLA	Constrained Optimization by Linear Approximation