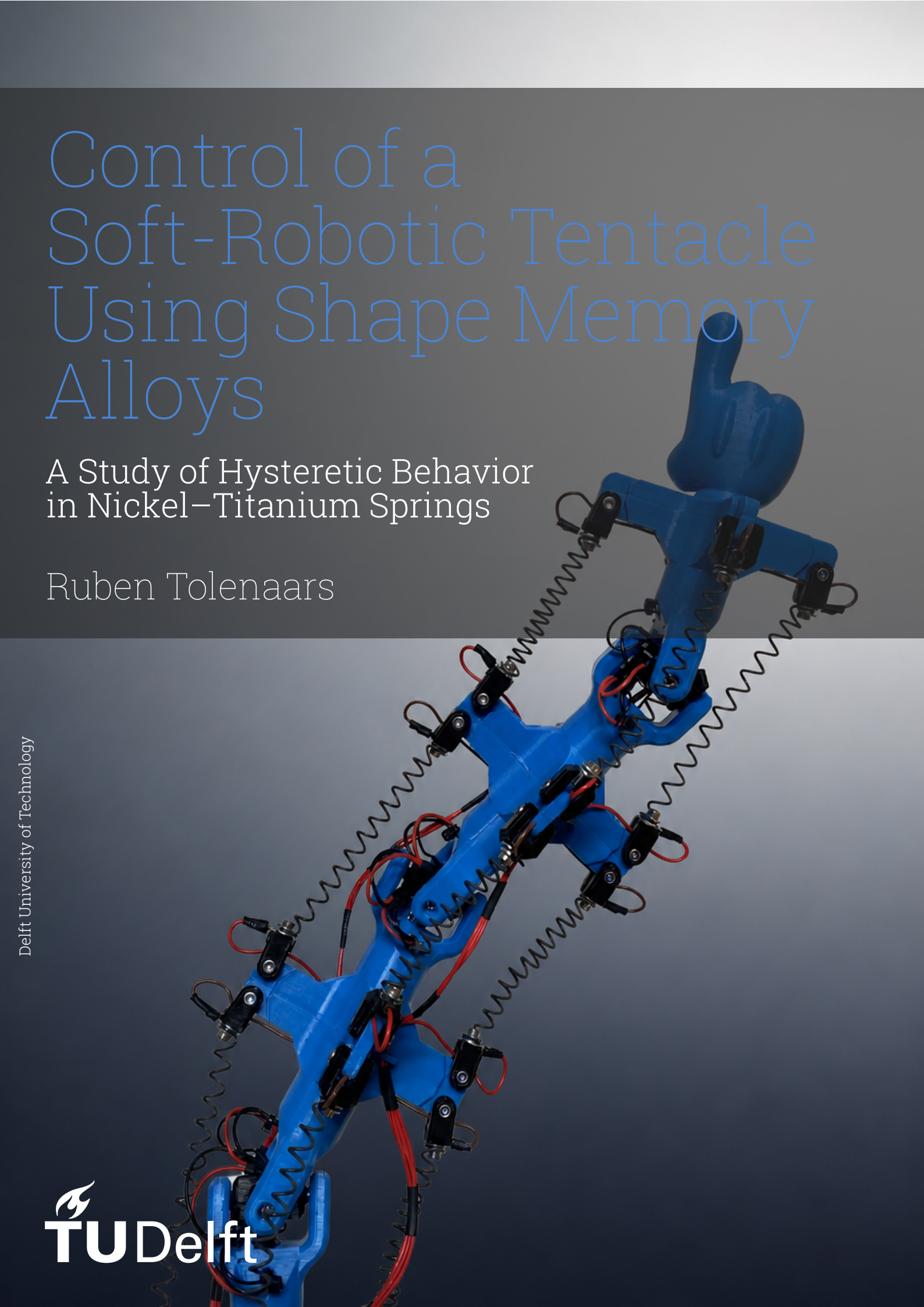
Control of a Soft-Robotic Tentacle Using Shape Memory Alloys

A Study of Hysteretic Behavior in Nickel–Titanium Springs

Ruben Tolenaars



Delft University of Technology

TU Delft

Control of a Soft-Robotic Tentacle Using Shape Memory Alloys

A Study of Hysteretic Behavior in Nickel–Titanium Springs

by

Ruben Tolenaars

Instructor:	J. Jovanova
Teaching Assistant:	Q. Chen
Project Duration:	October, 2025 - June, 2026
Faculty:	Faculty of Mechanical Engineering, Delft

Cover:	Original image created by the author, enhanced with AI-assisted editing tools
Style:	TU Delft Report Style

TU Delft

Preface

This thesis presents the design, control, and evaluation of a soft robotic tentacle actuated by shape memory alloys (SMAs). The work was carried out as part of the Master's programme in Mechanical Engineering at Delft University of Technology (TU Delft), and aims to explore the feasibility of using SMAs in biomimetic robotic systems.

The project focused on both the mechanical design of a prototype tentacle and the development of control strategies for SMA actuators. This included the design of a dedicated test bench, the implementation of different control approaches, and the experimental evaluation of system performance.

I would like to thank my supervisor, Jovana Jovanova, for her guidance and support throughout this project. Her feedback and insights were invaluable during the development of this research.

Ruben Tolenaars
Delft, June 2026

AI Statement

Artificial intelligence (AI) tools, such as ChatGPT and Google Gemini, were used to improve the flow, readability, spelling, and grammar of this work. These tools were not used to generate technical content or original ideas, but solely to refine the presentation of the text. All technical content, results, and conclusions presented in this thesis are based on the cited literature or the author's own work.

Summary

This research evaluated the feasibility of using SMAs in a biomimetic tentacle and investigated which control strategy is most suitable for this purpose. The literature review showed that SMAs have rarely been used to control the pose of a soft robotic tentacle. This gap in the literature was the main motivation for developing a prototype tentacle, which was ultimately used to evaluate both the performance of the SMAs and the implemented control strategy.

From the theoretical study of SMAs and their control methods, it was concluded that their thermo-mechanical response is highly non-linear and hysteretic. The literature further suggested that more advanced control strategies are typically required to achieve accurate control. In particular, many approaches rely on a feedforward–feedback (FF–FB) structure, where the feedforward effort is obtained from a model. For this reason, the Preisach hysteresis model was selected to describe the SMA behavior, with the aim of incorporating it into a controller. To identify this model and evaluate the FF–FB controller, a dedicated test bench with a single SMA actuator was developed.

Three different calibration procedures were performed to identify the Preisach model, which proved to be a time-consuming and delicate process. The resulting model was able to accurately predict major hysteresis loops, but performed poorly for minor loops. In addition, the hysteretic behavior of the SMA was found to be rate-dependent, which was not captured by the model. The model was implemented in the FF–FB controller and compared with a PID controller. It was concluded that the PID controller outperformed the FF–FB controller in most performance criteria, mainly due to its simplicity and the modeling errors present in the Preisach model.

Based on these findings, the PID controller was implemented in a prototype tentacle consisting of three independently rotating sections. Each section contained two orthogonal axes, each actuated by a pair of antagonistic SMAs. The controller showed consistent and reliable behavior, with reasonable tracking performance. However, further improvements in tracking accuracy would likely require a more advanced controller, as the PID controller does not account for the system dynamics.

Overall, it was concluded that using SMAs in a biomimetic tentacle is a feasible approach for applications where high accuracy and fast response are not required, and where simplicity, compliance, or silent operation are important criteria. Achieving higher performance, however, remains challenging due to the complex and hysteretic behavior of SMAs, and would require more advanced modeling and control strategies. Therefore, further research is needed to significantly improve upon the results presented in this work.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
SMA	Shape Memory Alloy
SME	Shape Memory Effect
SE	Superelastic Effect
PID	Proportional–Integral–Derivative (controller)
PI	Proportional–Integral (controller)
FF-FB	Feedforward-Feedback (controller)
PWM	Pulse-Width Modulation
FORC	First-Order Reversal Curve
SISO	Single-Input Single-Output
MISO	Multi-Input Single-Output
FOC	Field-Oriented Control
CAD	Computer-Aided Design

Symbols

Symbol	Definition	Unit
T	Temperature	[°C]
$\dot{T}$	Time derivative of temperature	[°C/s]
T_m	Measured temperature (thermocouple)	[°C]
T_r	Reference temperature	[°C]
T_{ff}	Feedforward temperature	[°C]
T_{fb}	Feedback temperature	[°C]
T_{sp}	Setpoint temperature	[°C]
ϵ	Strain	[-]
ϵ_r	Reference strain	[-]
ϵ_m	Measured strain	[-]
σ	Stress	[Pa]
σ_r	Reference stress	[Pa]
σ_m	Measured stress	[Pa]
$e_T(t)$	Temperature error signal	[°C]
$e_\epsilon(t)$	Strain error signal	[-]
R	Electrical resistance	[Ω]
I	Electrical current	[A]
V	Voltage	[V]
P	Electrical power	[W]
D	PWM duty cycle	[-]
$u(t)$	Control input (PWM duty cycle)	[-]
y	System output (strain)	[-]
α, β	Preisach switching thresholds	[°C]
$\mu(\alpha, \beta)$	Preisach weight function	[-]
$\gamma_{\alpha,\beta}$	Preisach hysteresis operator	[-]
$F(\alpha', \beta')$	FORC increment	[-]
$S^+(t)$	Region of active hysterons	[-]

Symbol	Definition	Unit
$\mathcal{T}(t)$	Temperature history	[°C]
ρ	Electrical resistivity	[$\Omega \cdot m$]
L	Length	[m]
A	Cross-sectional area	[m^2]

1 Introduction

1.1. Motivation and Background

Traditional robotic systems are typically constructed from rigid links and joints, resulting in mechanisms that are precise, but often limited in adaptability and safety when interacting with the environment. In contrast, soft robotics is an emerging field that focuses on the use of flexible materials—such as elastomers or alloys for actuation. The compliant structure of soft robots allows for a high, potentially infinite, number of degrees of freedom while ensuring safe interaction with the environment.

An interesting inspiration for such robots is the octopus. Octopus arm muscles are muscular hydrostats and operate under the isovolumetric principle [37, 12, 11], meaning that muscle tissue is essentially incompressible and the total volume remains constant during deformation. Therefore, a reduction in one dimension results in an expansion in another. An octopus arm consists of three main muscle groups composed of striated muscle cells: transverse, longitudinal, and oblique muscles [37, 12], as shown in Figure 1.1. The oblique muscle fibers are arranged in layers wound in both clockwise (CW) and counterclockwise (CCW) directions. Contraction of the longitudinal muscles shortens the arm, whereas contraction of the transverse muscles elongates it, resulting in antagonistic action between the two muscle groups. Activation of the CW and CCW oblique muscle layers produces twisting of the arm in the CCW and CW directions, respectively. Simultaneous contraction of the transverse and longitudinal muscles stiffen the arm, making it able to carry loads. If the longitudinal muscles do not contract uniformly around the arm's circumference, the arm bends toward the region with the least stiffness. Thus, the octopus arm can adapt its stiffness through three primary muscle groups, allowing it to extend, shorten, bend and twist in all directions.

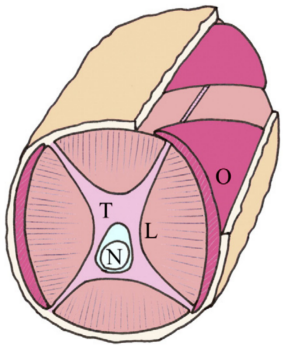


Figure 1.1: Cross-sections of an octopus arm showing muscular arrangement [12]. T: transverse muscle, L: longitudinal muscle, O: oblique muscle, N: central nerve cord.

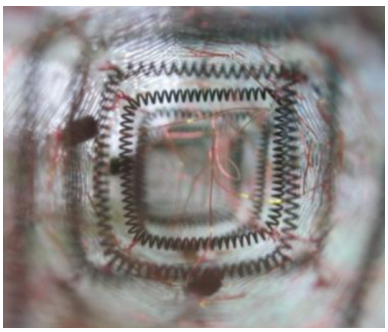
To develop a soft biomimetic tentacle, it is important to identify the most suitable type of artificial muscle. A literature review [36] was conducted to compare various artificial muscle technologies for their

applicability in a soft robotic tentacle, including pneumatic actuators, dielectric elastomers, and shape memory alloys (SMAs). The study concluded that nickel–titanium shape memory alloys, commonly referred to as *nitinol*, are among the most promising artificial muscles for soft robotic tentacles. This is primarily due to their high force density, compactness and safety.

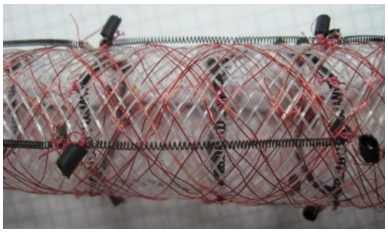
SMAs recover to a predefined shape upon heating above a certain transformation temperature, which is know as the *shape-memory-effect*. This behavior comes from a reversible solid-state phase transformation between the martensite and austenite crystal structures. When an SMA wire or spring is mechanically deformed in its low-temperature martensitic phase and subsequently heated, it contracts toward its original shape, thereby generating mechanical work. As a result, SMAs function as thermally driven actuators, with nitinol being the most extensively studied and widely available.

The main advantage of using SMAs in robotic applications instead of conventional actuators, such as electric motors, is that SMAs can provide a mechanically simple actuation mechanism. Whereas electric motors consist of multiple components, an SMA-driven robot can achieve motion using only a single element. Furthermore, SMA operation is completely silent, free of vibrations, relatively low cost, and capable of high force output. Therefore, implementing SMAs in a robotic tentacle could—in theory—be a simple and cost-effective solution for designing a robot with high dexterity.

A notable implementation of SMAs in a soft robotic tentacle identified in the literature study is the design presented in Figure 1.2. The tentacle is fully flexible, containing no rigid components, and closely mimics an octopus arm, with muscle orientations identical to the natural octopus arm. The artificial arm consists of a braided sleeve with two types of SMA actuators: radial springs (Figure 1.2a) and longitudinal springs (Figure 1.2b). Contraction of the radial springs causes the sleeve to elongate and its diameter to decrease, whereas contraction of the longitudinal springs shortens the sleeve and increases its diameter. Copper wires run parallel to the threads of the braided sleeve (Figure 1.2b), which minimizes their influence on the overall stiffness. To enhance performance, a silicone layer is wrapped around the sleeve and filled with mineral oil (Figure 1.2c). The oil improves cooling of the SMA springs—thus increasing their actuation bandwidth.



(a) Radial SMA actuators in the braided sleeve of the artificial octopus arm.



(b) Longitudinal SMA actuators with copper wires parallel to the braided sleeve.

(c) Complete arm with silicone layer around the braided sleeve, filled with mineral oil for cooling and protection.

Figure 1.2: Artificial octopus arm with (a) radial actuators, (b) longitudinal actuators, and (c) complete silicone-covered tentacle, adapted from [11].

Although the tentacle presented in Figure 1.2 is a promising concept, its performance is limited by the control strategy, which implements either full actuation or cooling of the SMA actuators, rather than driving the actuators to an intermediate state. Another example that uses a similar full-on/full-off strategy is the so-called snake arm robot depicted in Figure 1.3 [19], which consists of eight sections that rotate by actuating coiled SMA springs. SMA implementations similar to those shown in Figure 1.2 and Figure 1.3 exist in the literature; however, most of them rely on simple open-loop control strategies without proportional control.

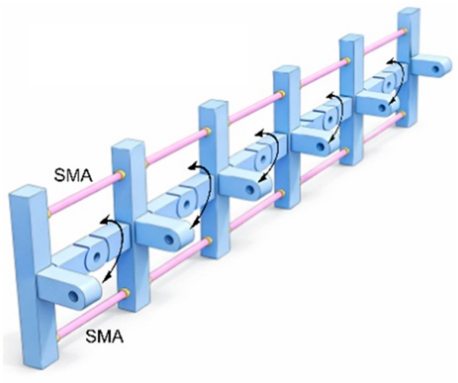


Figure 1.3: Structure of a SMA-actuated robot. Adapted from [32].

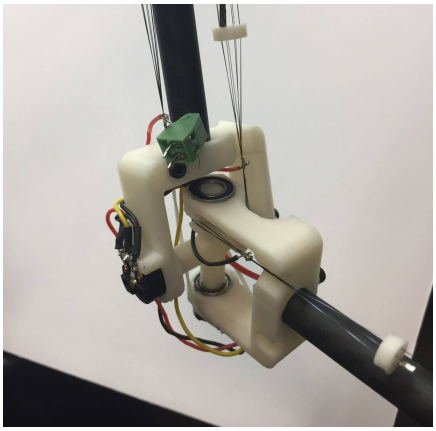


Figure 1.4: Joint of a ball-balancing robot. Adapted from [13].

Control applications have been studied extensively throughout the literature, where most methods focus on controlling a single SMA or a pair of antagonistic SMAs. Larger robotic structures that implement multiple actuators with proportional control are less commonly reported. An example of such a complete robotic structure is a balancing arm robot that implements a sliding mode controller to balance a ball on a paddle. One of the robot's joints, which contains two degrees of freedom and is controlled using bundled SMA wires, is depicted in Figure 1.4. This robot shows a surprisingly high actuation bandwidth of up to 2 Hz and, more importantly, demonstrates that SMAs can be used in a controlled environment.

1.2. Research Gap

Despite being promising concepts, the aforementioned SMA implementations lack certain properties in terms of controllability and simplicity. For example, the tentacle presented in Figure 1.2 shows a significant performance limitation: the contraction of its springs is not proportional. Instead, a simple full-on/full-off control strategy is used to actuate the SMAs. While this approach is sufficient to achieve basic bending motions, it limits positioning accuracy and restricts the tentacle's ability to perform smooth, controlled, and repeatable movements. The literature review on artificial muscle technologies [36] indicates that achieving such controlled behavior with SMAs is inherently challenging due to their non-linear thermomechanical response and hysteresis. Hysteresis is the phenomenon in which the state of the SMA depends not only on its current input, but also on its input history, thereby complicating proportional control.

Furthermore, the balancing arm robot depicted in Figure 1.4 does implement proportional control, but remains mechanically complex, since its movement mechanism relies on multiple components, such as bearings and bundled SMA wires. Although bundled SMA wires generally provide a large force output, their stroke is often small [32], which limits the achievable range of motion of the balancing arm robot. Therefore, this system lacks the high dexterity that could potentially be achieved with coiled SMA springs.

All in all, although extensive research has been conducted on SMA control, these methods have rarely, if ever, been implemented in biomimetic tentacle systems. Therefore, the objective of this research is to develop a robotic tentacle actuated by Nitinol springs with proportional control. Such a system could be valuable in applications requiring high dexterity and adaptability, including inspection in confined spaces, search and rescue operations, and human–robot interaction scenarios where safety and compliance are critical. Additionally, a robotic arm employing alternative actuation methods instead of conventional electric motors may provide a viable solution in applications where traditional motor-driven systems are unsuitable or impractical.

1.3. Research Questions and Goals

Although designing a fully soft biomimetic tentacle—such as the one presented in section 1.1—is an appealing concept, implementing proportional control in such a system is currently too challenging. The

complex geometry of the braided sleeve makes it difficult to predict the individual strains of the multiple SMA actuators required to achieve a desired tentacle deflection. Therefore, the primary objective of this research is to design and control a soft robotic tentacle composed of multiple discrete sections that can rotate with respect to each other. Each individual section—of which an example is provided in Figure 1.5—contains three or more SMA actuators that can be actuated to create a deflection angle between that section and the one connected to it. Using this approach, the controller design becomes more manageable, as the focus can be placed primarily on control development rather than on the complexities of a fully continuous structure. Furthermore, the control performance can be evaluated at the level of a single section, or even a single SMA actuator, which simplifies analysis and validation compared to considering the entire tentacle system.

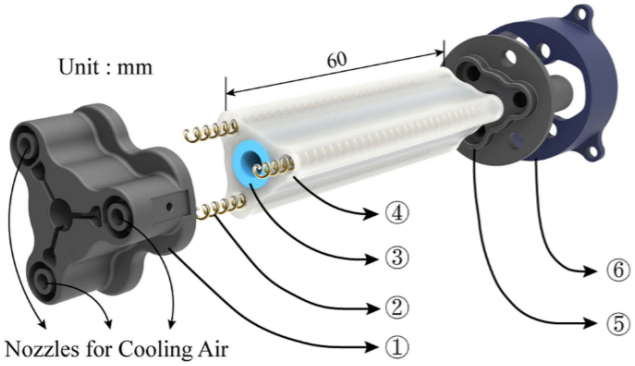


Figure 1.5: Example of a single tentacle section that can be used to construct a full tentacle by connecting the top cap of one section to the bottom cap of another. (1) rigid top cap; (2) SMA actuators (×3); (3) inner soft body; (4) outer soft body; (5) rigid bottom cap; (6) connection base. Actuation of the SMAs results in a relative deflection between the top and bottom caps. Adapted from [5].

Thus, the first research goal is to design and construct a prototype, desktop-scale soft robotic tentacle composed of multiple actuated sections. Few, if any, such robotic tentacles have been reported in the literature, making this an interesting area to explore. The main challenges associated with this goal include selecting suitable materials, designing the mechanical structure (e.g., how the sections rotate relative to each other), determining the required electronics to drive the SMAs, and developing a method to estimate the angles between sections.

The second research goal is to develop a control strategy capable of controlling the strain of SMA actuators, thereby enabling proportional control of each section. Although various control strategies exist in the literature, they have rarely been implemented in larger-scale biomimetic tentacles. It is therefore interesting to evaluate the achievable tracking performance in terms of tracking error, settling time, bandwidth, stability, and consistency. In addition, this research considers the implementation complexity of the controller, the required tuning effort, and its flexibility and adaptability. The main challenge of this design goal is to identify a suitable control strategy that can handle the non-linear and hysteretic behavior of SMAs while meeting these performance criteria.

The third research goal is to evaluate the feasibility of using SMAs as actuators in a soft biomimetic tentacle with respect to controllability and performance. Feasibility is determined by several factors. First, it depends on the ability of the controller to regulate the strain and thereby control the pose of the tentacle. Second, the use of SMAs is considered feasible if there are potential engineering applications in which such actuators offer a solution. Finally, feasibility is influenced by the potential for future performance improvements and the effort required to achieve them.

Consequently, the research goals and their associated research questions can be summarized as follows:

1. **Design and construction of a prototype**
   Design and construct a desktop-scale soft robotic tentacle composed of multiple actuated sections.

- How can such a prototype be designed and which materials are required?
- What electronics are required to control the system?
- How can the state of the system be estimated?

2. **Development of a control strategy**
   Develop a control strategy capable of controlling the strain of SMA actuators, enabling proportional actuation of each section.
   - Which control strategy is most suitable to achieve good tracking performance?
   - What is the tracking performance in terms of tracking error, settling time, bandwidth, stability, and consistency?
   - What is the implementation complexity, and how much effort is required for tuning?
3. **Feasibility evaluation of SMAs**
   Evaluate the feasibility of using SMAs as actuators in a soft biomimetic tentacle with respect to controllability and performance.
   - Can the pose of the tentacle be accurately controlled with the proposed controller?
   - For which engineering applications could SMAs be suitable actuators in such a system?
   - What is the potential for future performance improvements, and how much effort would this require?

1.4. Thesis Outline

An overview of the fundamental theory of SMAs, existing control strategies, and a model describing the thermomechanical response of SMA actuators is provided in subsection 2.1.1, section 2.2, and section 2.3, respectively. Based on this, it is concluded that developing an experimental setup for modeling and controlling a single SMA—rather than an entire tentacle—makes achieving the overall research objective more manageable. Therefore, a test bench for the modeling and control of a single SMA actuator is described in section 2.4. In chapter 3, the model calibration method, the performance of the model, and the performance of two control strategies—a PID controller and a feedforward–feedback controller—are evaluated for a single actuator using this test bench. Based on these results, it is concluded that a PID controller is a more promising solution for implementation in a complete tentacle design. This complete tentacle design, including an analysis of its performance, is presented in chapter 4. Finally, the research goals and research questions are revisited and answered in the conclusion provided in chapter 5.

2

Shape Memory Alloy Theory and Design Methodology

To design a biomimetic tentacle composed of SMAs, it is important to first understand their fundamentals and thermomechanical behavior, which is described in section 2.1. Section 2.2 explores the existing control strategies found throughout the literature. Based on this, it is concluded that a model is required for accurate strain control. The theory of a widely used model for SMAs—known as the Preisach model—is described in section 2.3. The experimental setup intended for identification of the Preisach model, as well as for control of a single actuator, is described in section 2.4. Finally, section 2.5 provides a summary of this chapter.

2.1. Shape Memory Alloy Fundamentals

2.1.1. Properties of Shape Memory Alloys

SMAs are a class of smart materials capable of returning to their original shape after deformation. In recent decades, they have gained increasing popularity across various engineering fields such as aerospace [27], biomedical, civil, automotive, and aeronautical engineering due to two unique properties: the shape memory effect (SME) and the superelastic effect (SE), also referred to as the pseudoelastic effect [9]. The SME is the ability to "memorize" a predefined shape, to which the material returns upon heating. The SE is the ability to undergo large deformations (5–10%) and fully recover from them without exhibiting plastic deformation [31, 11, 9, 14, 4].

These unique properties enable SMAs to function as actuators. At low temperatures and in an unloaded condition, SMAs exist in the *twinned martensite* state [31, 4, 9]. When subjected to a sufficiently large tensile stress, they transform into the *detwinned martensite* state. This deformation remains even after the stress is removed. Upon heating above a critical temperature, the SMA undergoes a phase transformation to *austenite*, causing the material to return to its original shape without any plastic deformation due to the SME. A typical thermomechanical characteristics curve is shown in Figure 2.1, illustrating the following behaviors:

1. Starting from zero strain at low temperature, the material undergoes elastic deformation while remaining in the *twinned martensite* phase.
2. The stress is increased, plastic deformation occurs and the material transforms into *detwinned martensite*.
3. When the applied stress is removed, the material retains its *detwinned martensite* phase along with the residual strain.
4. The material is heated up to austenite start temperature (A_s) and transition to *austenite* begins.
5. At austenite finish (A_f), the material fully transitions to *austenite* phase. A reduction of strain is observed.

6. Upon cooling to room temperature, *austenite* transforms back to *twinned martensite*, while the strain remains unchanged.

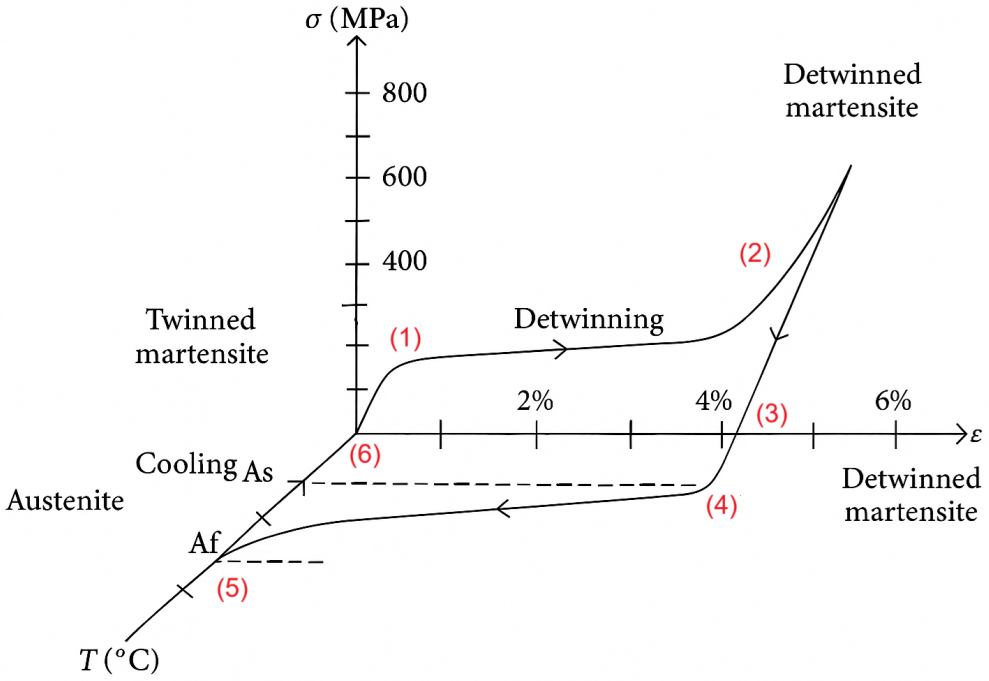
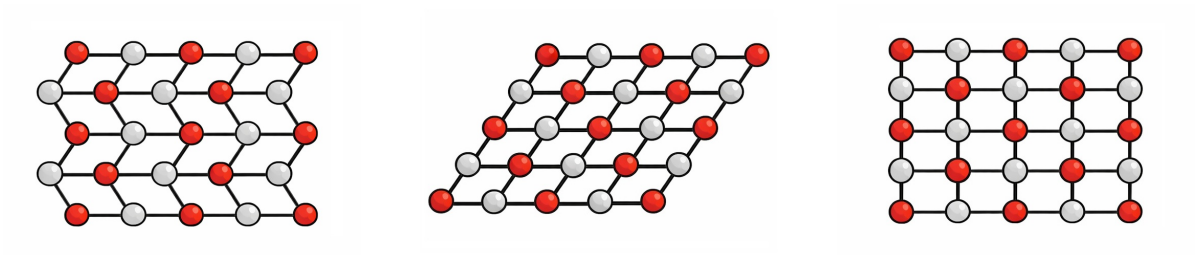


Figure 2.1: Thermomechanical characteristics curve of a typical SMA material, adapted from [31].

At the atomic level, the three distinct crystal lattice structures are shown in Figure 2.2. In *twinned martensite* (present under no load at low temperatures), the crystal lattice forms a mirror-image structure. When mechanical stress is applied, the lattice undergoes a process called detwinning, where the atomic planes reorient to align with the load, creating deformed martensite. The material retains this deformed shape even after the stress leaves the material. Upon heating above the transformation temperature, the alloy undergoes a solid-state phase change into *austenite*, recovering its original shape by rearranging the atoms back into a rigid, highly symmetric cubic lattice [31].



(a) Twinned Martensite **(b)** Detwinned Martensite **(c)** Austenite

Figure 2.2: Three phases of shape memory alloys: (a) Twinned Martensite undeformed at cold temperature, (b) Detwinned Martensite under deformation at cold temperature, and (c) Austenite at high temperature.

The first shape memory alloy was discovered by Ölander in 1932, based on a Gold-Cadmium alloy [29]. Today, the most widely used and researched SMAs are nickel-titanium alloys, commonly known as Nitinol. Nitinol actuators dominate the market and are the standard choice in most commercial applications [9, 31]. Heating of such actuators is most commonly achieved by passing an electric current through the material, also known as Joule heating. In some applications, such as the millimeter-scale crawling robot described in [14], alternative heating methods like combustion are used, as they do not require a heavy and large battery. However, for precise control applications, as required for a

soft robotic tentacle, Joule heating is more practical since the current through the material can be more easily controlled.

2.1.2. Hysteresis in Shape Memory Alloys

Although SMA actuators are promising for use in soft robotics, arguably their largest drawback is the hysteretic stress–strain–temperature relationship. Hysteresis is the phenomenon in which the state of a system depends not only on its current input but also on its history, preventing the system from returning to its original state along the same path. The hysteretic behavior of SMAs results from their solid-state phase transformation cycles. Within the transition temperature range, the SMA exists in a hybrid state, where part of the material is in the martensitic state and part is in the austenitic state.

This hysteretic behavior is visualized in Figure 2.3. The red curve shows the major hysteresis loop of an SMA actuator at a constant stress level. A major hysteresis loop occurs when the SMA undergoes a complete cycle between its fully martensitic and fully austenitic states. This requires the material to be heated above the austenite finish temperature (A_f) and cooled below the martensite finish temperature (M_f), ensuring no residual phases remain from the previous cycle [9, 31, 14]. In the figure, the SMA is heated (red curve) from the saturated martensite state to the saturated austenite state and then cooled back to the saturated martensite state. This curve is highly non-linear and shows different paths for heating and cooling due to hysteresis. The dashed blue lines in Figure 2.3 represent four minor hysteresis loops, which occur when the SMA is heated or cooled from a hybrid, non-saturated state. An infinite number of minor loops can exist within the major hysteresis loop. Moreover, the shape of the hysteresis loops depends not only on temperature but also on the applied load, as illustrated in Figure 2.4, which shows five major hysteresis loops under different stress levels. Therefore, besides hysteresis and non-linear behavior, the strain control method must also account for varying stress levels.

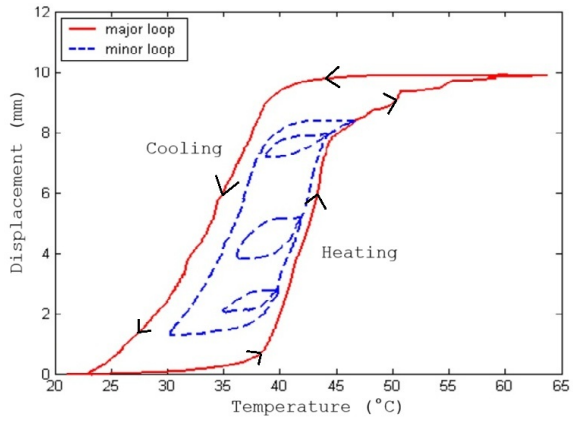


Figure 2.3: Displacement–temperature major hysteresis loop (red) and minor hysteresis loops (blue) at a constant stress level. Adapted from [20].

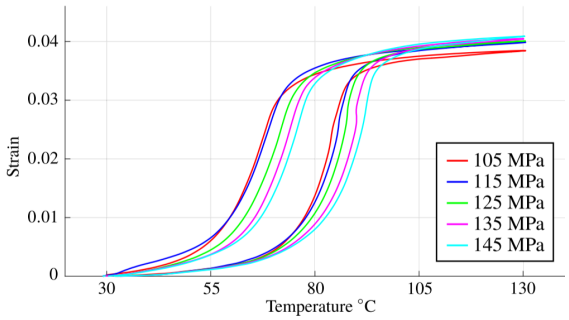


Figure 2.4: Strain–temperature major hysteresis loops of a 127 μm Nitinol SMA wire at five different stress levels. Reproduced from [14].

Another notable property of SMAs is their strong and repeatable correlation between strain and resistance. The electrical resistance of the actuator changes with its elongation, allowing it to measure its own deformation without the need for an external sensor. For this reason, SMAs are often referred to as “self-sensing” [31, 21]. The resistance (R) varies with length (L), cross-sectional area (A), and resistivity (ρ), as shown in Equation 2.1 [21].

$$R = \frac{\rho L}{A} \tag{2.1}$$

If the alloy remains in a single phase, this relationship can be assumed to be linear. However, since heating or cooling induces a solid-state phase transformation, the resistivity of the SMA changes depending on the fraction of austenite and martensite within the material. Because the formation of

martensite and austenite fractions is hysteretic and non-linear, the strain–resistance relationship is likewise hysteretic and non-linear. Figure 2.5 shows the major hysteresis loop of this relationship at a constant stress level.

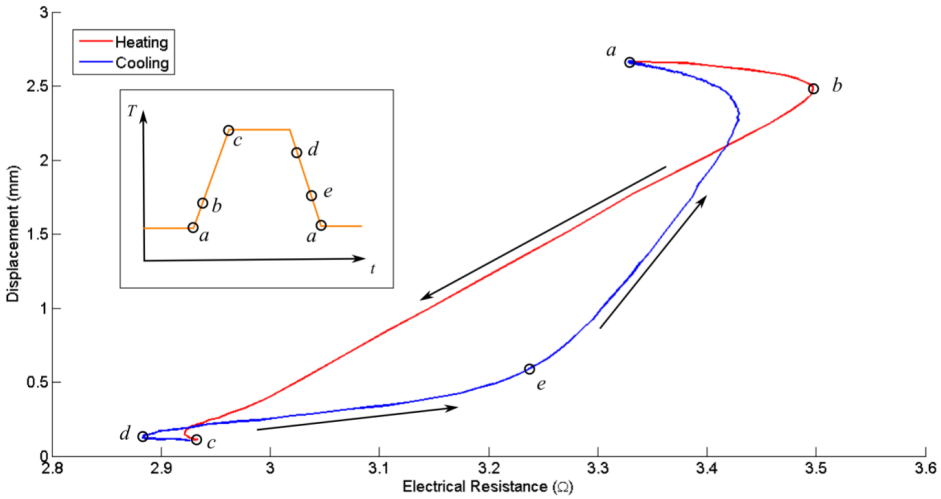


Figure 2.5: Major hysteresis loop of the strain–resistance relationship for a Nitinol wire during a full loading–unloading cycle. The inset illustrates an approximate evolution of the wire temperature throughout the transformation. Adapted from [21].

Furthermore, the strain–resistance relationship depends not only on the austenite and martensite fractions, and therefore on the temperature, but also on the applied stress to the material [21]. Consequently, distinct hysteresis loops can be expected depending on the load applied to the actuator. Thus, although the strain–resistance relationship can be exploited to measure the actuator's strain without external sensors, this comes with the challenge of accurately modeling this behavior, thereby complicating the control strategy of the robotic tentacle.

2.2. Control of Shape Memory Alloys

To address the research question of how to control the strain of SMAs, this section evaluates existing control strategies reported in the literature. The objective of the controller is to accurately track a reference strain through heating and cooling of the actuator. As stated in section 2.1, heating is typically achieved through Joule heating by passing an electric current through the actuator. Cooling is generally achieved through natural convection to the surrounding air, although other methods exist to enhance cooling rates, such as conductive cooling using mineral oil, as illustrated in Figure 1.2 in the previous chapter. Since the compliant nature of SMA actuators makes them sensitive to stress fluctuations, the controller must account for variations in stress. Therefore, the control objective is to regulate the actuator strain under time-varying stress conditions.

Numerous control strategies for regulating the strain or displacement of SMAs have been reported in the literature. Among these, PID-based control strategies are widely used due to their simplicity and effectiveness. However, variants of the conventional PID controller are often employed, as the standard PID controller is linear and may therefore be inadequate for capturing the non-linear behavior of SMAs. An example of such an approach is presented in [1], where the gains of a PID controller are tuned in real time within a closed-loop displacement control scheme using fuzzy logic. Another approach, described in [35], employs an adaptive PID controller to enhance the performance of the standard PID scheme. The implementation described in [38] introduces a bilinear term to compensate for the non-linear behavior of the SMA. This is achieved by passing the PID output through a non-linear scaling function before applying it to the actuator. In this approach, the output strain of an SMA $y(k)$ is assumed to depend on the strain at the previous sampling instant $y(k-1)$, the control input u, and the coefficients a, b, and η, as follows:

$$y(k) = -ay(k-1) + b\left[1 + \frac{\eta}{b}y(k-1)\right]u(k-1) \tag{2.2}$$

The output of this system can be linearized by introducing an additional input v and redefining the control input as:

$$u(k-1) = \frac{1}{1 + k_b\, y(k-1)}\, v(k-1) \tag{2.3}$$

Substituting Equation 2.3 into Equation 2.2, the resulting expression yields the linearized model:

$$y(k) = -ay(k-1) + b(v-1) \tag{2.4}$$

The non-linear scaling function used in the control scheme described in [38] to obtain the linearized plant is given in Equation 2.5, where k_b is a tuning parameter and $y_{ref}(k)$ is the reference strain. As illustrated in Figure 2.6, the PID controller output $v(k)$ is multiplied by this compensator to produce the final control signal $u(k)$. Note that the control scheme reduces to a conventional PID controller when k_b is set to zero. This implementation demonstrates that a conventional PID controller can be extended to handle a non-linear system by incorporating a single tunable gain k_b, thereby preserving the simplicity and structure of the standard PID approach.

$$\frac{1 + k_b\, y_{\text{ref}}(k)}{1 + k_b\, y(k-1)} \tag{2.5}$$

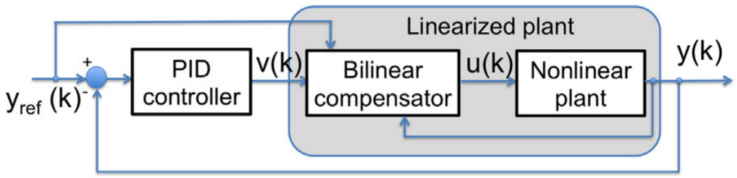


Figure 2.6: Control architecture using a bilinear compensator to linearize the plant for PID control. Here, y_{ref} is the reference strain, y is the measured actuator strain, v is the output of the PID controller, and u is the final controller output after applying the bilinear gain. Reproduced from [38].

Although PID control is widely used for SMAs, many control strategies reported in the literature use a hysteresis model to improve tracking performance. The most common approaches are empirical models based on the Preisach model, modified Preisach models, or the generalized Prandtl–Ishlinskii model to capture hysteresis effects [17]. These strategies are typically implemented as feedforward–feedback (FF-FB) schemes, where the feedforward component compensates for hysteresis and the feedback controller—often a PI or PID controller—improves tracking accuracy and rejects disturbances. Other approaches also exist, such as the dynamic model described in [17], which accounts for stresses induced by solid-state phase transformations to estimate actuator forces and accelerations. This method employs a dynamic thermomechanical model consisting of four components: a phase transformation model, a heat transfer model, a constitutive law model, and a dynamic model, which together are used for controller design. However, since most control methods in the literature rely on static empirical models, this research focuses exclusively on control strategies based on such static implementations.

The control architectures described in [23, 40, 18] all employ an empirical model in the feedforward signal to compensate for hysteresis, combined with a PID controller in the feedback signal to handle disturbances, achieving good tracking performance. In [40], the output of the inverse Prandtl–Ishlinskii hysteresis model generates a desired temperature signal that is fed into a temperature controller regulating the actuator. The feedback component is implemented as a PI controller, which uses the strain error to produce a corrective signal that is added to the feedforward term. Similarly, the methods in [23,

18] adopt a FF-FB control scheme. This general approach is illustrated in Figure 2.7. Note that the hysteresis model must be inverted to implement this control method, as the feedforward term needs to generate a temperature signal that—when applied to the actuator—produces the reference strain ϵ_r. In its non-inverted form, the Preisach model outputs strain based on the temperature history and current stress, making it unsuitable for control applications.

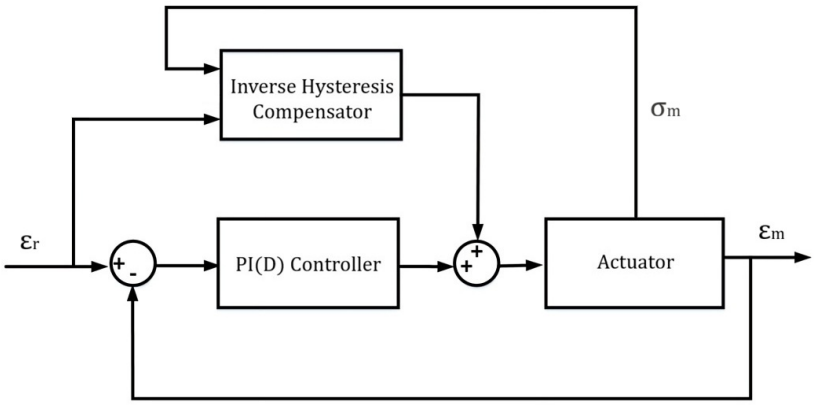


Figure 2.7: Generalized control scheme employing an inverse hysteresis model in the feedforward path and a PID controller in the feedback path. Here, ϵ_r denotes the reference strain, σ_m the measured actuator stress, and ϵ_m the measured strain.

A drawback of the two control methods discussed so far is that they require direct measurement of the actuator strain. As mentioned in subsection 2.1.2, strain can be estimated from the actuator's electrical resistance. However, because resistance is hysteretic and depends on the fraction of austenite and martensite in the material, this approach is beyond the scope of this research. For similar reasons, other methods in the literature rely on encoders or lasers to measure SMA displacement. What distinguishes the sensor requirements for our application is the need to monitor multiple actuators simultaneously. This introduces potential complications in the tentacle design, as adding multiple sensors increases wiring complexity and can complicate the overall system integration.

To eliminate the need for strain sensors, the method proposed by [14] measures the actuator temperature instead of strain, and compares this measured temperature to the output of an inverted Preisach model within a feedforward control scheme, as illustrated in Figure 2.8. In this approach, the reference strain ϵ_r and reference stress σ_r are fed into the inverted Preisach model, denoted Γ^{-1}, to generate the reference temperature T_r. To compensate for disturbances, T_r and σ_r are tracked using two independent closed-loop controllers tuned by the gains K_t and K_s.

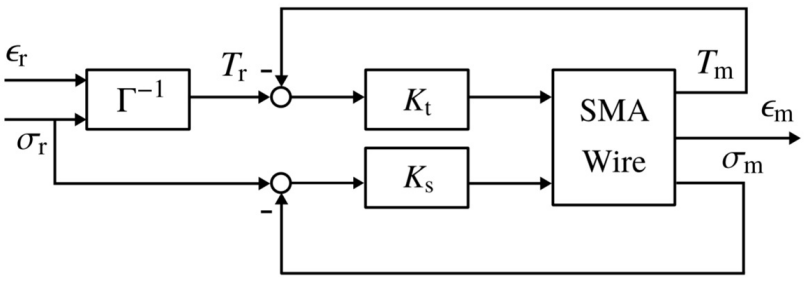


Figure 2.8: Feedforward control scheme for tracking the strain ϵ_m of an SMA actuator without using a strain sensor. The reference temperature T_r is computed by the inverse Preisach model Γ^{-1} from the reference strain ϵ_r and reference stress σ_r. The gains K_t and K_s are used to tune the independent feedback loops for the measured temperature T_m and stress σ_m, respectively. Reproduced from [14].

For implementation in a soft robotic tentacle, this approach may be more practical, as it relies on es-

timating the temperature of each actuator rather than directly measuring strain. By considering the history of current inputs to an actuator and making assumptions about heat transfer to the environment, the actuator temperature can be estimated without using a sensor. For example, by employing a single thermocouple and observing its temperature change as a function of input power and environmental heat transfer, it can be assumed that all SMAs in the tentacle behave similarly, allowing their temperatures to be estimated. The drawback of this method is that it requires an accurate temperature model.

To conclude this section, it is found that many existing control strategies employ a hysteresis model as a feedforward term in the controller, thereby compensating for the inherent hysteresis and non-linear behavior of SMAs. It may be possible that a PID controller will suffice for the final biomimetic tentacle. However, as suggested in the literature, PID control often results in poor tracking performance due to its linear nature. Therefore, a more advanced controller integrating a hysteresis model may be required to achieve good tracking performance. For this reason, the next section describes the Preisach hysteresis model that can be implemented within a control scheme.

2.3. Modeling of Shape Memory Alloys

Numerous hysteresis models for SMAs exist, which can be roughly divided into empirical models—such as the Prandtl–Ishlinskii model [40] or the Preisach model—which rely on experimental data to identify the system's behavior, and theoretical models—such as the thermomechanical model described by [17]—which are often based on a phase transformation model, heat transfer model, constitutive law model, and kinematic/dynamic model to predict the system's behavior. Among these, the Preisach model is by far the most widely used and reported modeling approach in the literature, which is why it is adopted in this research.

The classical Preisach model of hysteresis [24] is used to describe single-input single-output (SISO) systems. The fundamental building blocks of the Preisach model are hysteresis operators called *hysterons* [24, 1, 16, 14]. A schematic representation of such a hysteron ($\gamma_{\alpha\beta}$) is shown in Figure 2.9. The output of a hysteron depends on the threshold temperature parameters α and β, as well as on the temperature history $\mathcal{T}(t_i)$. For a given time t_i, the temperature history is represented by the ordered sequence $\mathcal{T}(t_i)$ defined in Equation 2.6.

When the input temperature exceeds α, the hysteron switches to the "on" state and its output becomes 1. When the temperature decreases below β, the hysteron switches to the "off" state and the output becomes 0. Within the interval $\beta \leq T(t) \leq \alpha$, the output remains equal to its previous state, which results in the memory effect of the hysteron. Its output can therefore be summarized as shown in Equation 2.7.

$$\mathcal{T}(t_i) = \{T_1, T_2, \ldots, T_i\}, \qquad T_i = T(t_i) \tag{2.6}$$

$$\gamma_{\alpha,\beta}[T] = \begin{cases} 1, & \text{if } T(t) > \alpha, \\ 0, & \text{if } T(t) < \beta, \\ \text{previous state}, & \text{if } \beta \leq T(t) \leq \alpha. \end{cases} \tag{2.7}$$

According to the Preisach model, the strain output of an SMA actuator is the summation of a continuum of weighted hysteresis operators $\gamma_{\alpha\beta}$, each characterized by unique temperature thresholds α and β. The weights of the hysterons must be empirically calibrated through experimental measurements. Intuitively, a single hysteron can be interpreted as representing a fraction of the material, where the "on" state corresponds to that fraction being in the austenitic phase, and the "off" state corresponds to it being in the martensitic phase. Therefore, when the SMA is in a hybrid state consisting of both austenite and martensite, multiple hysterons with different temperature thresholds collectively represent the material's internal phase distribution, forming the fundamental principle of the Preisach model.

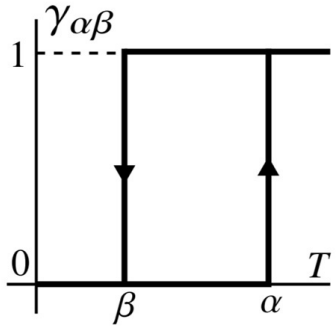


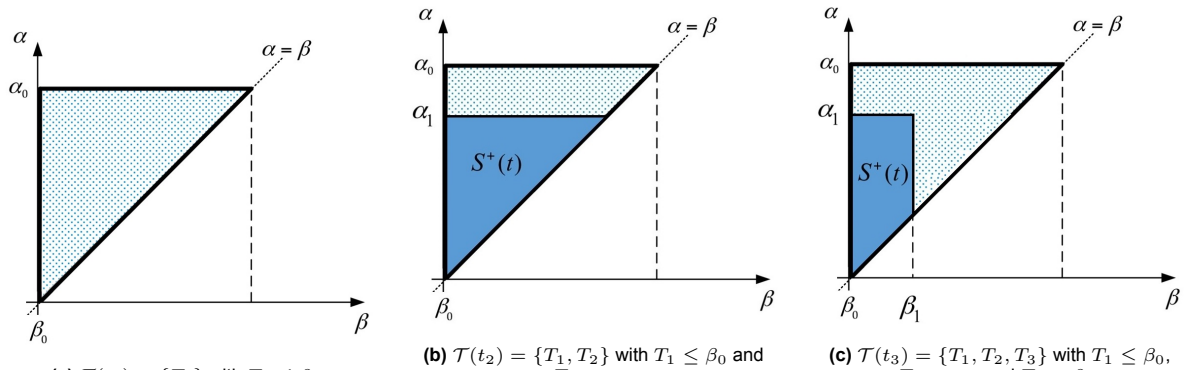
Figure 2.9: Schematic representation of a Preisach hysteron characterized by threshold parameters α and β, with $\alpha > \beta$, and temperature input (T). The output state switches to 1 when the input T exceeds the upper threshold α, and switches to 0 when the input falls below the lower threshold β. Within the interval $\beta < T < \alpha$, the hysteron retains its previous state, thereby exhibiting hysteretic memory. Adapted from [14].

As the temperature varies over time, each hysteron adjusts its output according to its temperature thresholds. The overall output of the Preisach model is the superposition of all hysterons, each weighted by an arbitrary function $\mu(\alpha, \beta)$, known as the Preisach function or weight function [1, 14]. For constant stress levels, the continuous Preisach strain function is given in Equation 2.8. It should be noted that the value of $\gamma_{\alpha,\beta}[T](t)$ depends not only on the current temperature $T(t)$ but also on the entire temperature history $\mathcal{T}(t_i)$.

$$\epsilon(t) = \iint_{\alpha \geq \beta} \mu(\alpha, \beta)\, \gamma_{\alpha,\beta}[T](t)\, \mathrm{d}\alpha\, \mathrm{d}\beta \tag{2.8}$$

The saturation temperatures—that is, the temperatures at which the SMA is fully in the martensitic or austenitic state—must be determined empirically, as they depend on the alloy composition and stress level [14]. They are denoted as α_0 and β_0, representing the upper saturation bound for austenite and the lower saturation bound for martensite, respectively. Within these saturation bounds, an evenly spaced set of hysterons can be defined over the triangular region enclosed by the lines $\alpha = \alpha_0$, $\beta = \beta_0$, and $\alpha = \beta$. This geometric representation is known as the Preisach plane and is shown in Figure 2.10. When starting from a cold temperature $T_1 \leq \beta_0$, the SMA is in the saturated martensitic state and the temperature history becomes $\mathcal{T}(t_1) = \{T_1\}$. All hysterons defined within the triangular region are in the ”off” state, as shown in Figure 2.10a. If the temperature is subsequently increased to α_1, all hysterons with $\alpha < \alpha_1$ switch to the ”on” state, represented by the blue area in Figure 2.10b, and a new temperature maximum α_1 is established. The region in the Preisach plane corresponding to hysterons in the ”on” state is denoted as $S^+(t)$. When the temperature is then decreased to a new minimum β_1, a portion of $S^+(t)$ is removed from the Preisach plane, showing the wiping-out effect of the model. The set $S^+(t)$ can be interpreted as the fraction of the SMA in the austenitic state, while the wiping-out effect represents the fraction of material transforming back from austenite to martensite. Using this geometrical interpretation, the strain can also be computed as:

$$\epsilon(t) = \iint_{S^+(t)} \mu(\alpha, \beta)\ \mathrm{d}\alpha\, \mathrm{d}\beta \tag{2.9}$$



(a) $\mathcal{T}(t_1) = \{T_1\}$ with $T_1 \leq \beta_0$.

(b) $\mathcal{T}(t_2) = \{T_1, T_2\}$ with $T_1 \leq \beta_0$ and $T_2 = \alpha_1$.

(c) $\mathcal{T}(t_3) = \{T_1, T_2, T_3\}$ with $T_1 \leq \beta_0$, $T_2 = \alpha_1$, and $T_3 = \beta_1$.

Figure 2.10: Preisach plane with equidistant hysterons represented as blue dots. The temperature history $\mathcal{T}(t_n)$ is defined in Equation 2.6. The parameters α_0 and β_0 denote the upper and lower temperature bounds corresponding to austenite and martensite saturation, respectively. (a) The process starts at a low temperature $T_1 \leq \beta_0$, where the SMA is fully in the martensitic state. (b) The temperature is increased to a new maximum $T_2 = \alpha_1$, generating the region $S^+(t)$ (blue), which represents the hysterons in the "on" state. (c) The temperature is subsequently decreased to $T_3 = \beta_1$, reducing $S^+(t)$ and establishing a new minimum. Adapted from [18].

Each time $T(t)$ exceeds a previous maximum, that maximum is wiped out from the model and replaced by the new maximum; similarly, when $T(t)$ falls below a previous minimum, that minimum is also wiped out. Consequently, as $T(t)$ alternates between n successive maxima α_k with $k \in \{1, \ldots, n\}$ and $n-1$ minima β_k with $k \in \{1, \ldots, n-1\}$ satisfying $\alpha_k < \alpha_{k-1}$ and $\beta_k > \beta_{k-1}$—such that the α_k are sorted in decreasing order and the β_k in increasing order—a staircase $L(t)$ is formed in the Preisach plane. The curve $L(t)$ separates the regions $S^+(t)$ and $S^0(t)$, representing the hysterons in the "on" and "off" states, respectively, as illustrated in Figure 2.11. Furthermore, the region $S^+(t)$ can be subdivided into $n-1$ trapezoids Q_k and a small triangle $\Delta(\alpha_n, \beta_{n-1})$ defined by the lines $\alpha = \alpha_n$, $\beta = \beta_{n-1}$, and $\alpha = \beta$.

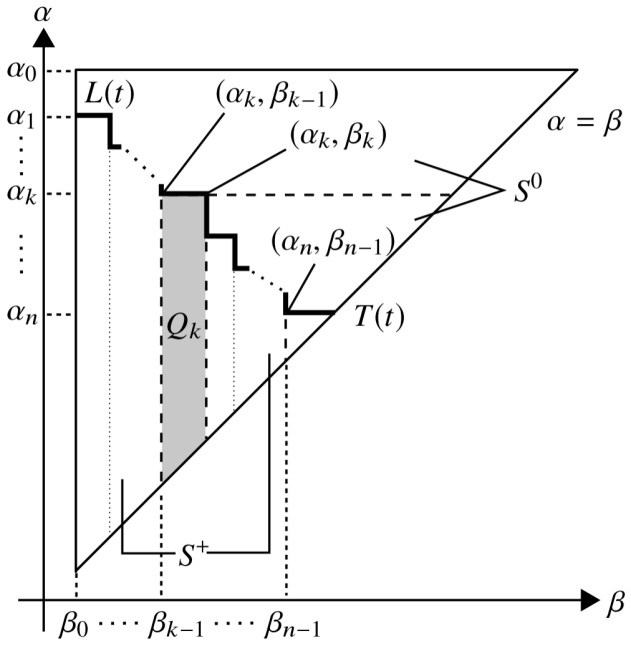


Figure 2.11: Schematic representation of the Preisach plane showing the variables used in the model. α_0 and β_0 denote the austenite and martensite saturation temperatures, respectively. The staircase $L(t)$ is defined by n temperature maxima α_k and $n-1$ temperature minima β_k, separating the hysterons in the "on" state—represented by the region S^+—from those in the "off" state—represented by the region S^0. The region S^+ is composed of trapezoids Q_k and the small triangle $\Delta(\alpha_n, \beta_{n-1})$, defined by the lines $\alpha = \alpha_n$, $\beta = \beta_{n-1}$, and $\alpha = \beta$. Reproduced from [14].

The continuous representation of the Preisach strain in Equation 2.8 is not suitable for a discrete-time

implementation. The method described in [14] introduces a discrete Preisach model that uses the staircase $L(t)$ as an alternative mechanism to store the system state. Since the weight function $\mu(\alpha, \beta)$ is an abstract mathematical concept that cannot be directly measured, the algorithm follows an approach that does not require the computation of $\mu(\alpha, \beta)$. Instead, a so-called *first-order-reversal-curve* (FORC) strain increment is used to identify the Preisach model and compute the strain. A FORC-increment is obtained by monotonically heating the actuator from its martensite saturation temperature β_0 to α' and then monotonically cooling to β'. Intuitively, this represents the strain "released" by the actuator when cooling from α' to β', assuming the actuator started from fully saturated martensite. Mathematically, a FORC-increment is expressed in Equation 2.10, where $f_{\alpha'}$ is the strain at temperature α' and $f_{\alpha',\beta'}$ is the strain after cooling from α' to β'.

$$F(\alpha', \beta') = f_{\alpha'} - f_{\alpha',\beta'} \tag{2.10}$$

This can be represented geometrically as shown in Figure 2.12. A FORC-increment corresponds to the region in the Preisach plane denoted by $\Delta(\alpha', \beta')$, defined as the triangle enclosed by the lines $\alpha = \alpha'$, $\beta = \beta'$, and $\alpha = \beta$. Thus, $\Delta(\alpha', \beta')$ represents the portion of the Preisach plane that is wiped out when the temperature decreases from $\alpha = \alpha'$ to $\beta = \beta'$, and it is equivalent to the surface integral of the weight function over the region in which the hysterons switch off after this temperature reversal. A FORC-increment can therefore be expressed as

$$F(\alpha', \beta') = \iint_{\Delta(\alpha',\beta')} \mu(\alpha, \beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta \tag{2.11}$$

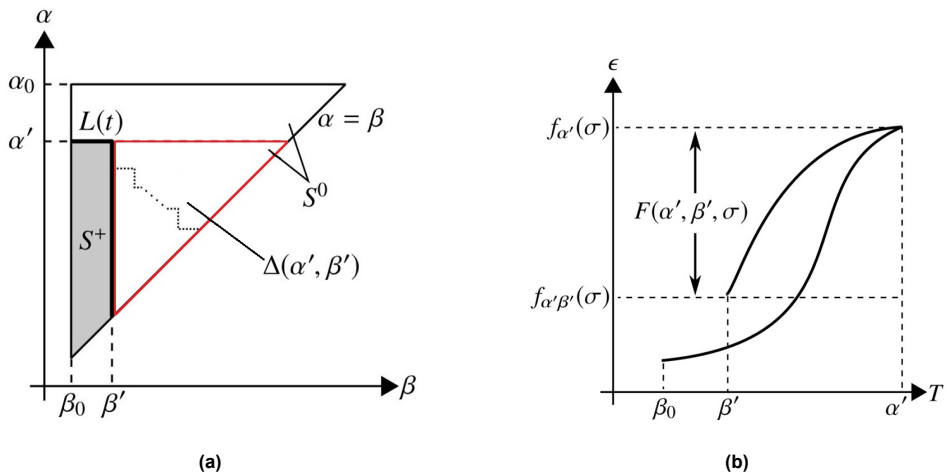


Figure 2.12: Geometrical interpretation of the *first-order-reversal-curve* (FORC) strain increment. **(a)** α_0 and β_0 denote the austenite and martensite saturation temperatures, respectively. The red triangle $\Delta(\alpha', \beta')$ represents the region that is wiped out from the Preisach plane when the temperature is increased from β_0 to α' and subsequently decreased from α' to β'. This process generates the staircase $L(t)$, which separates the regions S^+ and S^0, representing the hysterons in the "on" and "off" states, respectively. **(b)** Strain–temperature relationship of a FORC-increment with reversal-point temperature α' and end-point temperature β'. The strain increment between $f_{\alpha'}$ and $f_{\alpha',\beta'}$, denoted as $F(\alpha', \beta')$, corresponds to the surface integral of the weight function over the region where the hysterons switch off after the first temperature reversal, that is, the triangle $\Delta(\alpha', \beta')$. Reproduced from [14].

Since $S^+(t)$ equals the sum of all trapezoids $Q_k(t)$ and the small triangle $\Delta(\alpha_n, \beta_{n-1})$, Equation 2.9 can be rewritten as the sum of the surface integrals of the weight function $\mu(\alpha, \beta)$ over the trapezoids $Q_k(t)$ and the surface integral over the triangle $\Delta(\alpha_n, \beta_{n-1})$:

$$\epsilon(t) = \iint_{S^+(t)} \mu(\alpha, \beta)\ \mathrm{d}\alpha\, \mathrm{d}\beta = \sum_{k=1}^{n(t)-1} \iint_{Q_k(t)} \mu(\alpha, \beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta + \iint_{\Delta(\alpha_n,\beta_{n-1})} \mu(\alpha, \beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta \tag{2.12}$$

Furthermore, $Q_k(t)$ can be expressed as the difference between two overlapping triangles, that is, two FORC-increments. The grey-shaded region $Q_k(t)$ in Figure 2.11 corresponds to the triangle enclosed by $\alpha = \alpha_k$, $\beta = \beta_{k-1}$, and $\alpha = \beta$, which is equal to $F(\alpha_k, \beta_{k-1})$, minus the triangle enclosed by $\alpha = \alpha_k$, $\beta = \beta_k$, and $\alpha = \beta$, which is equal to $F(\alpha_k, \beta_k)$. Therefore, the surface integral of $\mu(\alpha, \beta)$ over a trapezoid $Q_k(t)$ can be written as:

$$\iint_{Q_k(t)} \mu(\alpha, \beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta = F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k) \tag{2.13}$$

Similarly, the small triangle $\Delta(\alpha_n, \beta_{n-1})$ in Figure 2.11 can be expressed as:

$$\iint_{\Delta(\alpha_n, \beta_{n-1})} \mu(\alpha, \beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta = F(\alpha_n, \beta_{n-1}) \tag{2.14}$$

By substituting Equation 2.13 and Equation 2.14 into Equation 2.12, the discrete Preisach strain output becomes:

$$\epsilon(t) = \sum_{k=1}^{n(t)-1} \left[F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\right] + F(\alpha_n, \beta_{n-1}) \tag{2.15}$$

Since this expression contains measurable FORC-increments, the Preisach model can be identified empirically. By collecting a finite number of equally spaced FORC-increments within the Preisach plane, the continuous FORC surface can be approximated using linear interpolation. The surface formed by the collection of FORC-increments fully characterizes the Preisach model and can be considered a hypersurface. Further details on the system identification process are provided in chapter 3.

It must be noted, however, that Equation 2.15 only holds for monotonically increasing temperatures, i.e., during heating. For decreasing temperatures, the last term $F(\alpha_n, \beta_{n-1})$ becomes zero, since it represents the small triangle $\Delta(\alpha_n, \beta_{n-1})$, which does not exist during cooling. In that case, the Preisach plane consists entirely of trapezoids $Q_k(t)$. Therefore, for monotonically decreasing temperatures, the strain output is given by Equation 2.15 without the last term.

Additionally, the strain definition in Equation 2.15 assumes that the model is *single-input-single-output* (SISO), as it considers only a single input (temperature) and a single output (strain). However, as discussed in subsection 2.1.2, the strain output of SMAs is *multi-input-single-output* (MISO), since it also depends on the applied stress. Consequently, the method described in [14] identifies separate FORC surfaces for different stress levels and linearly interpolates between two surfaces when the actuator stress falls between those levels. The FORC-increment definition in Equation 2.10 can therefore be updated to include the actuator stress σ, as shown in Equation 2.16. Similarly, Equation 2.15 can be modified to account for stress, and the final expression for strain, including time-varying stress and the sign of the temperature derivative, is provided in Equation 2.17.

$$F(\alpha', \beta', \sigma) = f_{\alpha'}(\sigma) - f_{\alpha', \beta'}(\sigma) \tag{2.16}$$

$$\epsilon(t, \sigma) = \begin{cases} \displaystyle\sum_{k=1}^{n(t)-1} \Big[F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\Big] + F(\alpha_n, \beta_{n-1}), & \text{if } \dot{T}(t) > 0 \text{ (heating)}, \\ \displaystyle\sum_{k=1}^{n(t)} \Big[F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\Big], & \text{if } \dot{T}(t) < 0 \text{ (cooling)}. \end{cases} \tag{2.17}$$

2.4. Experimental Setup

Rather than directly constructing the robotic tentacle with multiple sections, a test bench was developed for modeling and control of a single actuator for the following reasons:

- The behavior of a single SMA actuator is significantly less complex than that of multiple interacting actuators, making analysis and modeling more manageable.
- The actuation electronics and sensors can be tested and validated on a small scale before integrating multiple units into a full tentacle design.
- The test bench provides an experimental platform to identify the Preisach model described in section 2.3.
- It enables the implementation and evaluation of the control strategies presented in section 2.2 in a simple experimental setting.

.

2.4.1. Actuation Test Bench

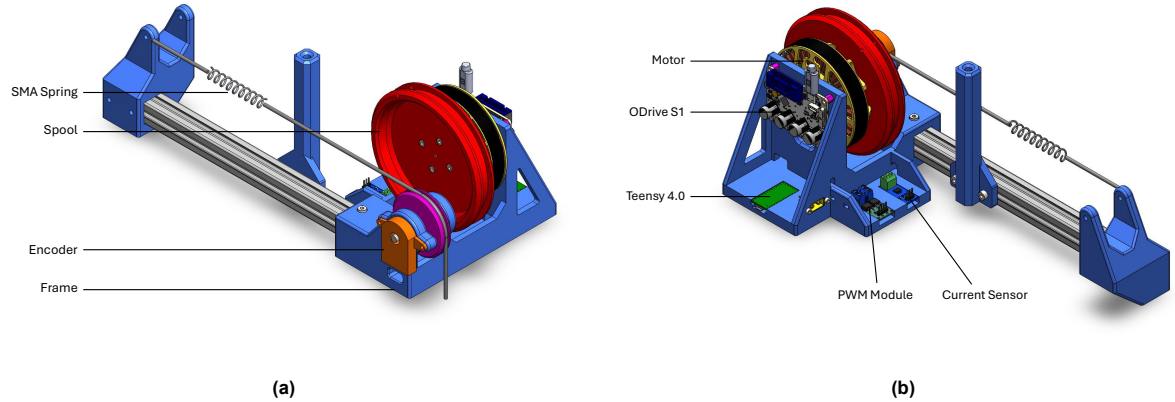


Figure 2.13: CAD representation of the SMA test bench without thermocouple. (a) Front view showing the SMA spring, spool, CUI AMT102-V capacitive encoder, and supporting frame. (b) Electronic and actuation components, including the Eaglepower 8308 90KV permanent magnet motor, ODrive S1 motor driver, Teensy 4.0 microcontroller, PWM module, and ACS712 breakout board current sensor.

The test bench—illustrated in Figure 2.13—was designed using CAD and is primarily composed of 3D-printed components. It provides two distinct methods for applying a load to the SMA spring, and therefore supports two different testing approaches. In the first method, the SMA spring is connected to a rope that passes over a 3D-printed wheel (pink), whose shaft is coupled to an encoder. By attaching a mass to the rope, a constant force is applied to the actuator, while the resulting strain is estimated from the encoder measurements. This method is primarily intended for model identification, as the calibration of the Preisach model (see chapter 3) requires strain measurements under constant load conditions. In the second method, the rope is attached to a large red spool connected to a motor. The motor is driven by an ODrive S1 motor driver [28], which implements *field-oriented control* (FOC) to generate a desired torque. This configuration enables the application of a time-varying load by commanding a time-varying motor torque, and is therefore particularly useful for evaluating controller performance under dynamic loading conditions.

The actuator strain is defined as a function of the displacement $u(x)$ as shown in Figure 2.14. The displacement $u(x)$ is obtained from the encoder measurements, taking into account the diameter of the pink wheel shown in Figure 2.13. Since the displacement must be referenced to a well-defined configuration, the reference state is chosen as the saturated austenitic state (high temperature). Accordingly, the displacement is set to $u(x) = 0$ when the actuator is fully transformed to austenite. The corresponding spring length in this configuration is defined as L_0. The maximum possible extension of the actuator is defined as the spring length in the saturated martensitic state (low temperature), limited to the maximum admissible extension that does not cause permanent plastic deformation of the SMA

spring. With this definition, the strain is formulated such that it increases with increasing temperature. Although the physical length of the spring decreases during heating, this choice ensures that the strain definition is consistent with most conventions in the literature. The strain ϵ is therefore defined as:

$$\epsilon = \frac{\text{max extension} - u(x)}{L_0} \tag{2.18}$$

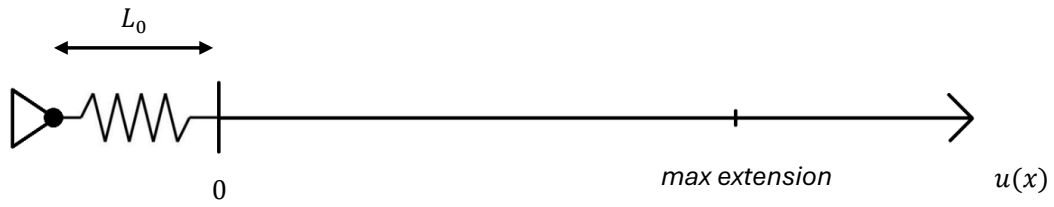


Figure 2.14: Schematic of the SMA actuator showing the variables used to define the actuator strain: L_0 is the spring length at saturated austenite, $u(x)$ is the actuator displacement relative to the austenitic reference, and the maximum extension corresponds to the spring length in saturated martensite without causing plastic deformation.

2.4.2. Electronics and Sensors

PWM Module

For actuating the SMA within the test bench, a breakout board containing two AOD4184A MOSFETs [3] is used to generate a *pulse-width-modulation* (PWM) signal. When the input from the microcontroller is set to high, the MOSFETs close the circuit between the power supply and the actuator, causing the SMA to heat up. When the input is low, the circuit opens and no heating occurs. The duty cycle is defined as the ratio between the “on” time and the total period of the signal, and is given by:

$$D = \frac{t_{on}}{T} = \frac{t_{on}}{t_{on} + t_{off}} \tag{2.19}$$

The average current is defined as the integral of the current over one period, divided by the length of that period:

$$I_{avg} = \frac{1}{T} \int_0^T i(t)\, dt \tag{2.20}$$

Since the current through the SMA is equal to the input voltage V_{in} divided by its resistance R, this expression can be written as:

$$I_{avg} = \frac{1}{T} \left[\int_0^{t_{on}} \frac{V_{in}}{R}\, dt + \int_{t_{on}}^T 0\, dt \right] \tag{2.21}$$

Evaluating the integral yields:

$$I_{avg} = \frac{V_{in}}{R} \frac{t_{on}}{T} = \frac{V_{in}}{R} \cdot D \tag{2.22}$$

Therefore, Equation 2.22 shows that the average current through the actuator can be controlled by adjusting the duty cycle. Since the actuator temperature increases with electrical current due to Joule heating, the temperature can therefore be regulated by varying the duty cycle.

It should be noted, however, that this relationship is derived under the assumption that the SMA behaves as a purely resistive load without inductive effects. Furthermore, as discussed in section 2.1, the resistance R of the SMA is hysteretic and strongly dependent on temperature and phase fraction, and is therefore not constant. As a result, the true relationship between duty cycle and average current—and consequently temperature—is not strictly linear, as suggested by Equation 2.22.

Furthermore, to minimize switching losses—and consequently the voltage drop—the PWM module operates at a relatively low frequency of 1000 Hz. Since losses primarily occur during switching transitions, this frequency ensures the MOSFET is either fully on or fully off for over 99% of the time, making switching losses negligible.

Current Sensor

To monitor the current through the actuator—and for system identification purposes that will be clarified in chapter 3—the test bench is equipped with a current sensor. The selected sensor is a low-cost and widely available breakout board based on the ACS712 Hall-effect IC [2], which provides an analog output voltage proportional to the measured current. Since the current through the actuator is a PWM signal, the sensor output is likewise PWM-modulated. To obtain the average current, a running average is computed in software using 2000 samples. Further details on this method, including its associated sources of error, are provided in subsection 2.4.4.

Temperature Sensor

Since the Preisach model described in section 2.3 requires accurate temperature control for model identification, a temperature sensor is necessary to implement a closed-loop temperature controller. In most applications reported in the literature, thermocouples are used for this purpose. In this setup, a Unit KMeter K-type digital thermocouple is selected, offering a resolution of 0.25°C, an accuracy of ±2%, and a maximum sampling rate of 10 Hz [22]. The primary reason for choosing this sensor is its electrically isolated I^2C interface, which provides isolation between the thermocouple measurement side and the host (I^2C) side. This isolation protects the measurement circuitry from ground loops and electrical noise. An earlier implementation using a low-cost, non-isolated sensor resulted in unusable measurement data once the PWM module was activated, due to electrical interference. Replacing it with the KMeter resolved these issues and provided stable temperature measurements.

The hot junction—i.e., the tip—of the thermocouple is tightly wrapped around the SMA actuator. The temperature measurement therefore relies on conductive heat transfer between the actuator and the hot junction. As a result, a temperature difference exists between the actuator and the thermocouple tip, introducing an error in the estimated actuator temperature. To improve thermal conductivity, some methods—such as the approach described in [14]—use conductive oil to improve heat transfer between the actuator and the thermocouple junction, thereby reducing the temperature difference and corresponding measurement error. For simplicity, the described test bench does not employ such conductive oil.

ODrive and Motor

As explained in the previous section, the ODrive S1 and motor can be used to apply varying loads—such as step inputs—to the actuator in order to evaluate controller performance. Since the ODrive implements *field-oriented control* (FOC), it is capable of regulating the motor's position, velocity, or—most importantly for this application—its torque using an integrated closed-loop controller. The ODrive includes a magnetic encoder capable of sensing rotating magnetic fields, allowing the motor position to be measured by mounting a magnet on the motor's output shaft. Therefore, a magnet is installed between the motor and the ODrive (not visible in Figure 2.13). By measuring the motor position, the displacement—and therefore the strain—of the SMA actuator can be estimated, providing a way to evaluate the tracking performance of the controller under time-varying loads. Further details on the magnetic encoder are available in the ODrive documentation [28]. The motor used in the test bench—the Eaglepower 8308—is a relatively low-cost hobby-grade motor. It was selected primarily due to its availability at the time of development.

Microcontroller

For acquiring sensor data and controlling both the ODrive and the PWM module, a Teensy 4.0 microcontroller was selected [30]. The Teensy can be programmed in C++ using the Arduino IDE [6], while

offering significantly greater performance than a standard Arduino board. It features a 600 MHz processor, a larger number of digital and analog I/O pins, and 12-bit analog input resolution, compared to the 10-bit resolution available on most conventional Arduino boards. In addition, the Teensy 4.0 provides multiple UART and I^2C interfaces, which are required for communication with the ODrive motor driver and the thermocouple module.

Encoder

The encoder used in the test bench is an encoder with a resolution of 8192 counts per revolution [33]. Since this is an incremental encoder, it does not provide absolute position information, but instead generates pulses as it rotates. By counting these pulses with the microcontroller, the rotation of the pink wheel shown in Figure 2.13 can be determined, from which the displacement—and consequently the strain—of the actuator can be estimated. Defining the linear displacement $u(x)$ in terms of the encoder angle ϕ_{enc} (in radians) and the radius of the pink wheel r, the displacement can be expressed as:

$$u(x) = r\,\phi_{enc} \tag{2.23}$$

Consequently, the displacement resolution Res_u is proportional to the angular resolution Res_ϕ:

$$Res_u = r\,Res_\phi \tag{2.24}$$

Substituting the radius of the pink wheel, the resolution of $u(x)$ becomes:

$$Res_u = r \cdot \frac{2\pi}{8192} = 25\,\text{mm} \cdot \frac{2\pi}{8192} \approx 0.0192\,\text{mm} \tag{2.25}$$

Battery and Voltage Divider

To power the actuator, the PWM module is connected to a lithium-ion battery composed of three Samsung INR18650-30Q cells in parallel, providing a nominal voltage of 3.7 V and a total capacity of 9 Ah. For estimating the actuator's power consumption—for reasons that will be clarified in chapter 3—the battery voltage V_{bat} is monitored using a voltage divider circuit, as illustrated in Figure 2.15. The output voltage of the divider, V_{out}, is connected to an analog input pin on the Teensy, allowing the microcontroller to estimate the battery voltage using its 12-bit analog-to-digital converter. Since the battery voltage ranges between 3.0 V and 4.2 V (depending on the state of charge) and the Teensy's analog pins are rated for a maximum of 3.3 V, the resistances R_1 and R_2 are chosen so that V_{out} never exceeds 3.3 V while still maximizing the voltage range and resolution. In this setup, $R_1 = 83.5\,\text{k}\Omega$ and $R_2 = 309\,\text{k}\Omega$ were selected; the large resistance values minimize current draw through the divider and avoid unnecessary battery discharge. The output voltage of a voltage divider is given by:

$$V_{out} = V_{bat}\frac{R_2}{R_1 + R_2} \tag{2.26}$$

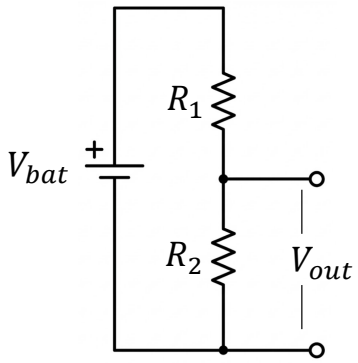


Figure 2.15: Schematic representation of the voltage divider circuit. V_{bat} is the battery voltage, V_{out} is the voltage measured by the microcontroller, and R_1 and R_2 are the chosen resistor values.

2.4.3. Control Architecture

The control architecture of the test bench is divided into two parts, as illustrated in Figure 2.16: a data acquisition and low-level control part using the Teensy, and a high-level system modeling and control part using Python on a desktop PC. The Teensy is responsible for reading sensor values, sending PWM signals to the PWM module, and controlling the ODrive. Additionally, the Teensy implements a closed-loop PI current controller for experimental purposes and a closed-loop PID temperature controller—which will be described in the next chapter—for identifying the Preisach model. The Teensy also filters the analog current and voltage divider measurements using a running average.

The Teensy sends the filtered current and battery voltage data, encoder position data, and thermocouple temperature data to the desktop PC at 40 Hz over serial communication running Python. Commands can also be sent from Python to the Teensy, including PWM, current, or temperature commands. In summary, the Teensy transmits sensor data to Python, while Python sends control commands to the Teensy. This architecture allows the entire test bench to be controlled and monitored through Python after the Teensy is programmed.

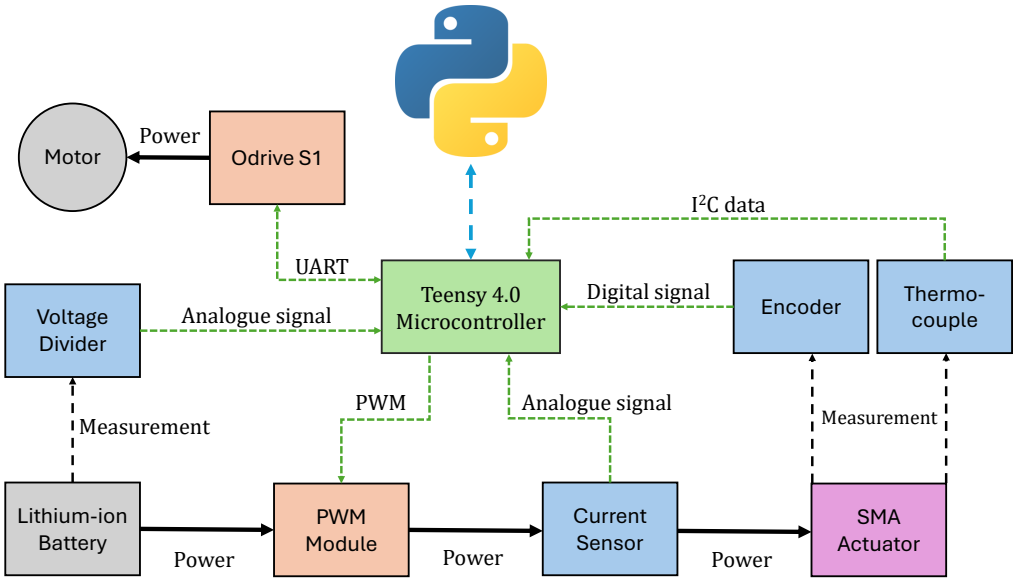


Figure 2.16: Schematic of the control architecture, showing the microcontroller (green) and Python interface including its sensors (blue), actuation components (orange), motor and battery (grey), and SMA actuator (pink).

2.4.4. Sources of Error and Noise

As explained previously, accurate temperature control is required to identify the Preisach model and construct the FORC surface described in section 2.3. An example of a temperature measurement consisting of 45 samples is shown in Figure 2.17a. Since the thermocouple includes integrated signal processing, it outputs temperature data to the microcontroller in discrete steps of 0.25 °C. The samples were recorded under a constant PWM signal with a duty cycle of 6.1 %, while keeping the actuator temperature at approximately 40 °C. Because the bandwidth of the KMeter thermocouple is below 10 Hz, a sampling frequency of 1 Hz was selected to ensure unique measurements.

As shown in Figure 2.17a, the thermocouple measurements contain noise that must be filtered to provide reliable temperature estimates for the controller. Therefore, the temperature readings are filtered using a running average defined as:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i] \tag{2.27}$$

where $y[n]$ is the filtered signal, N is the number of samples, and $x[n]$ represents the individual measurements. To determine the number of required samples that will provide a reliable temperature estimate, the sample mean, standard deviation and standard error of the mean are computed as shown in the following equations:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{2.28}$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{2.29}$$

$$\mathrm{SEM} = \frac{\sigma}{\sqrt{N}} \tag{2.30}$$

For control purposes, it is assumed that the error of the filtered temperature signal must remain within $-0.1\,^{\circ}\mathrm{C}$ and $+0.1\,^{\circ}\mathrm{C}$ of the true value, i.e., $\epsilon = 0.1\,^{\circ}\mathrm{C}$. Assuming normally distributed measurement noise, the 95% confidence interval is given by $\bar{x} \pm 1.96\,\mathrm{SEM}$. Therefore,

$$1.96 \frac{\sigma}{\sqrt{N}} \leq \epsilon \tag{2.31}$$

and solving for N yields

$$N \geq \left(\frac{1.96\,\sigma}{\epsilon} \right)^2 . \tag{2.32}$$

Using the experimentally determined standard deviation σ of the temperature measurements, the minimum number of samples required to achieve an uncertainty below $\pm 0.1\,^{\circ}\mathrm{C}$ was calculated using Equation 2.32, resulting in $N = 25$. This averaging window was therefore selected to filter the temperature signal according to Equation 2.27.

Similarly, the current measurements are filtered using a running average. A set of 100 current measurements sampled at 100 Hz, recorded while the actuator current equals zero, is shown in Figure 2.17b. Accurate current measurements are required to estimate the actuator input power, as will be discussed in chapter 3. Assuming $\epsilon = 0.01$, the required number of samples for filtering the current signal would be 85. However, because the current sensor measures a PWM-driven signal, its output alternates between approximately 0 A—when the PWM signal is low—and a higher current level—when the PWM signal is high. The true average current lies between these values and depends on the duty cycle.

One possible method to estimate the average current would be to multiply the measured high-level current by the duty cycle. However, since the Teensy's main control loop operates independently of the PWM timer, it cannot determine whether a specific sample was taken during the high or low phase of the PWM signal. To simplify implementation, a brute-force approach is implemented in which 2000 samples are used in the running average filter. Because the Teensy can sample the analog current sensor at very high rates (on the order of hundreds of kHz), this approach is feasible. Experimental validation showed that the filtered output remains stable to two decimal places, with variations confined to the third decimal place. An alternative method would be to use a hardware low-pass filter (e.g., a capacitor) to smooth the PWM signal before measurement. However, since the brute-force software approach proved reliable, it was implemented for simplicity.

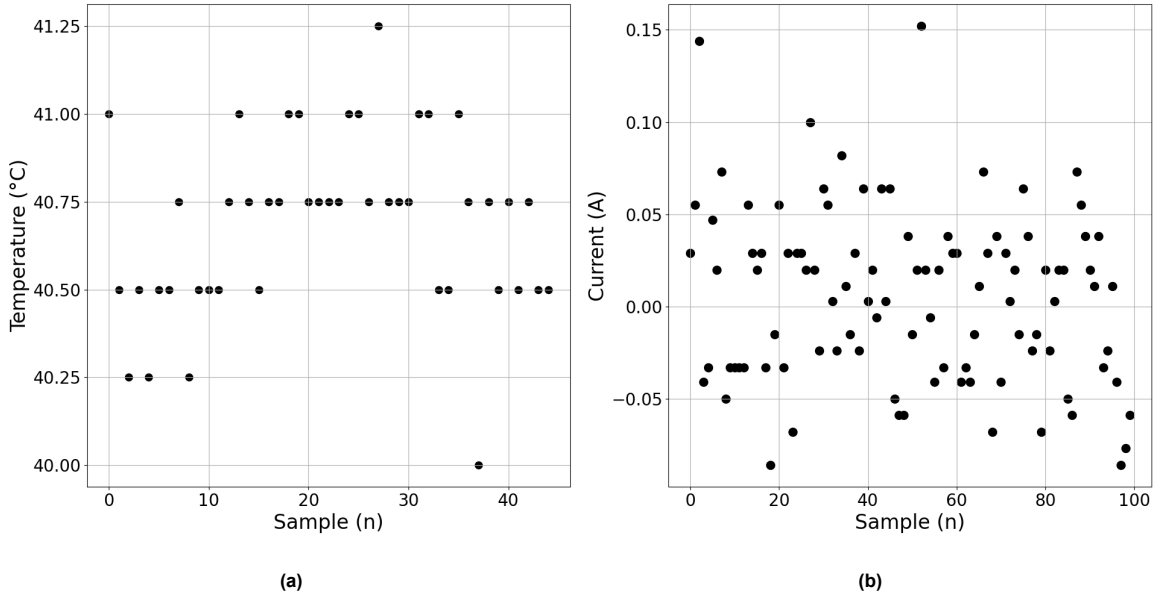


Figure 2.17: Scatter plots of measurement samples. (a) Temperature sensor data showing 45 samples recorded at 1 Hz while the actuator is heated using a PWM duty cycle of approximately 6.1 %. The sample mean is $\bar{T} \approx 40.683\,^{\circ}\mathrm{C}$ with a standard deviation of $\sigma_T \approx 0.252\,^{\circ}\mathrm{C}$. (b) Current sensor data showing 100 samples recorded at 100 Hz. The sample mean is $\bar{I} \approx 0.00544\,\mathrm{A}$ with a standard deviation of $\sigma_I \approx 0.0468\,\mathrm{A}$.

2.5. Chapter Summary

In summary, to control the strain of SMAs, the literature review indicates that a controller incorporating a hysteresis model will likely outperform a PID controller in terms of tracking performance, due to the strongly non-linear and hysteretic behavior of the actuator. Therefore, a Preisach hysteresis model, which can be identified by measuring FORC increments, will be implemented using the described test bench. To keep the analysis manageable, the modeling and control are initially performed on a single SMA actuator, while accounting for varying stress conditions. This approach provides a practical foundation for extending the control strategy to a multi-actuator biomimetic tentacle later on. The identification method, as well as the control performance of the implemented controllers for a single actuator, will be presented and evaluated in the next chapter.

3

Control of a Single SMA Actuator

The previous chapter explored the possibilities of using SMAs as actuators within a robotic tentacle composed of multiple (independent) sections. It was concluded that, to achieve good tracking performance, a more advanced controller is likely required due to the non-linear behavior of SMAs and their hysteretic thermomechanical properties. To progress toward the research goal of designing and controlling a complete tentacle, the control strategies discussed in this chapter are first evaluated on a single SMA actuator using the test bench described in section 2.4. Therefore, the research question of this chapter is formulated as follows: how can the strain of a single SMA actuator be controlled using a PID controller and a feedforward–feedback (FF–FB) controller, what level of performance can be achieved, and which strategy is most suitable for implementation in a robotic tentacle?

First, both control strategies and their underlying motivation are discussed in section 3.1. Since the FF–FB controller requires a hysteresis model—such as the one described in section 2.3—the identification procedure and performance evaluation of the Preisach hysteresis model are presented in section 3.2 and section 3.3, respectively. Subsequently, the performance and analysis of the PID controller are discussed in section 3.4, followed by those of the FF–FB controller in section 3.5. Based on the performance of both controllers, future improvements are discussed in section 3.6. To determine which control strategy is most suitable for implementation in a biomimetic tentacle, the controllers are compared and a final selection is made in section 3.7. Finally, a summary of this chapter is provided in section 3.8.

3.1. Control Strategy Overview and Motivation

The objective of the controllers described in the upcoming chapter is to control the strain of the SMA actuator in the test bench. As explained, the ultimate goal is to implement these controllers in a soft robotic tentacle, where the angles of the independent sections must be controlled. Three different types of controllers were highlighted in section 2.2: a PID controller with a non-linear gain compensator, a FF-FB controller consisting of a PID controller (feedback) and an inverse hysteresis compensator (feedforward), and a purely feedforward controller which does not require measurement of the actuator strain.

This chapter describes the design and implementation of only the PID controller and the FF-FB controller. The purely feedforward controller is not implemented, since the performance of the identified Preisach model can already be evaluated through the FF-FB controller. Implementing this controller would therefore not provide much additional academic contribution. Furthermore, as will become clear in section 3.5, the identified Preisach model contains errors that are too large for use in a purely feedforward scheme, meaning that some form of feedback is required to achieve accurate strain tracking. Due to time constraints, the tracking performance of both controllers was evaluated only for a fixed load of 2.5 N. Although the resulting strain differs for different loads, the thermomechanical behavior remains very similar, which will become clear in the upcoming chapter. For simplicity, the PID controller does not include a non-linear gain compensator due to the time constraints of this project. In addition, its

role is to serve as a baseline for comparison with the more advanced FF-FB controller.

3.2. Identification of the Preisach Model

3.2.1. Identification Procedure Based on Literature

As explained in section 2.3, the Preisach model is defined by a finite set of FORC increments that form a FORC surface. These FORC increments are equally spaced along the Preisach plane and must be empirically identified. By identifying a FORC surface for a selected set of constant stress levels, the Preisach model determines the strain for an intermediate stress level by interpolating between those stress levels. The martensite and austenite saturation temperatures, $\beta_0 = 25°\mathrm{C}$ and $\alpha_0 = 64°\mathrm{C}$, were determined experimentally. The Preisach plane is therefore defined over the triangular domain

$$\{(\alpha, \beta) \mid 25°\mathrm{C} \leq \beta \leq \alpha \leq 64°\mathrm{C}\}.$$

To discretize this domain, both the α- and β-axes were sampled with a uniform step size of $3°\mathrm{C}$. Only grid points satisfying $\alpha > \beta$ were retained, resulting in a triangular mesh over the the Preisach plane. With this discretization, 105 FORC measurements are required to identify a single FORC surface at a constant stress level.

Since the interval between β_0 and α_0 is discretized into 14 equidistant steps of $3°\mathrm{C}$, the identification of a single FORC surface requires 14 experiments. Each experiment corresponds to one reversal temperature α' on the Preisach grid. In each experiment, the following procedure is carried out:

- The actuator is heated from β_0 to a prescribed temperature α' on the discretized α-axis.
- The actuator is then cooled monotonically from α' to β_0.
- For every temperature $\beta' < \alpha'$ on the discretized grid, a FORC increment $F(\alpha', \beta', \sigma)$ is identified.

An example of such an experiment is shown in Figure D.1a, where all FORC increments are plotted for $\alpha' = 64°\mathrm{C}$ at a constant stress level. By performing all 14 experiments, i.e., for each $\alpha' \in [25, 64]°\mathrm{C}$ in steps of $3°\mathrm{C}$, the FORC surface shown in Figure D.1b is obtained. As explained above, this surface defines the Preisach model for a constant stress level. The strain — as given in Equation 2.17 — is calculated using bilinear interpolation whenever a required FORC increment does not lie exactly on a discretization point of the Preisach plane, which is the case in most real-time applications. Bilinear interpolation is performed by calculating a weighted average of the four nearest neighboring grid points in the two-dimensional (α, β) domain. The interpolated value is computed as

$$f(x, y) \approx f_{00}(1 - x)(1 - y) + f_{01}x(1 - y) + f_{10}(1 - x)y + f_{11}xy, \tag{3.1}$$

where f_{00}, f_{01}, f_{10}, and f_{11} denote the FORC increments at the four surrounding grid points of the rectangular cell containing the evaluation point. The normalized local coordinates $x, y \in [0, 1]$ are the relative position of the evaluation point within this cell, measured along the α- and β-directions. Therefore, the interpolated value represents the linearly weighted contribution of the four neighboring FORC increments according to the distance of the evaluation point from each grid node.

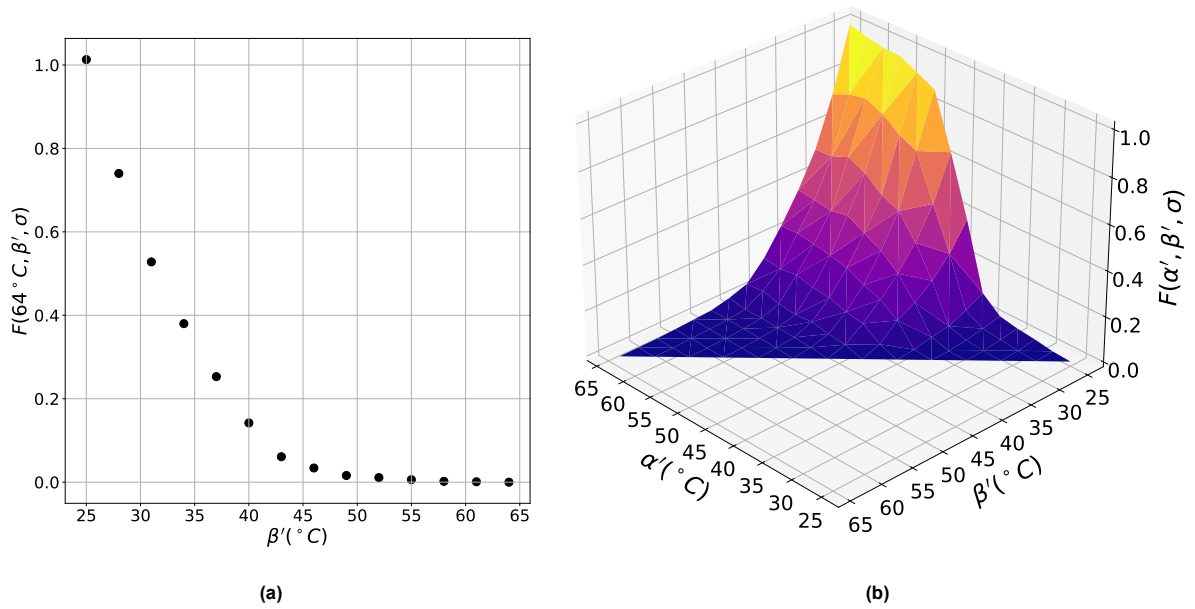


Figure 3.1: Examples of FORC increments. (a) Example of a FORC measurement with $\alpha' = 64\,^\circ\mathrm{C}$, showing a temperature reversal that yield s 14 FORC increments. (b) Complete set of FORC increments forming the surface that defines the Preisach model.

It is important to note that significant time and effort were spent on obtaining the FORC surfaces. In the first approach—described in Appendix D and following the method of [14]—the FORC increments were measured at temperature reversal points using a PI temperature controller. See Appendix C for the full design and tuning procedure of this controller. Using this temperature-control-based calibration, the resulting Preisach model showed very poor performance, with oscillatory behavior and physically unrealistic hysteresis loops. It was ultimately concluded that drift in the thermocouple caused the controller to drive the temperature non-monotonically toward unintended setpoints, resulting in incorrect FORC increments. The complete calibration procedure and failure analysis are provided in Appendix D.

To address the poor performance of the Preisach model obtained from the temperature-control-based calibration, a second calibration method based on PWM actuation was implemented. By applying a PWM signal with a constant duty cycle, the actuator reaches an unknown but stable temperature. Since PWM-based actuation does not rely on a closed-loop control scheme—but instead on the stable hardware PWM of the microcontroller—the actuator can be heated without large temperature fluctuations. During post-processing, the temperature at each PWM reversal point was estimated using thermocouple readings, encoder data, and a temperature model based on a steady-state energy balance. The resulting model showed substantial improvements compared to the one obtained using temperature-control-based calibration. The complete calibration procedure, temperature model, and model evaluation are provided in Appendix E.

3.2.2. Expanded Identification Procedure

As discussed in subsection 3.2.1, the Preisach model identified using the PWM-based calibration approach showed improved results. However, the model performed well during cooling trajectories—that is, when the actuator temperature decreases monotonically—but poorly during heating trajectories. Analysis of this model, presented in Appendix E, revealed that this is caused by the calibration procedure: since the FORC increments were obtained during cooling, the model does not capture the actuator behavior during heating. For this reason, the Preisach model is extended to a dual formulation, containing both a FORC surface for cooling and a FORC surface for heating for each applied load.

As explained, the identification of a cooling FORC surface requires multiple experiments. Each experiment contains multiple temperature reversals in which the FORC increments are identified. In such

a temperature reversal, the actuator is heated from the martensite saturation temperature β_0 to a prescribed temperature α' on the discretized Preisach grid and subsequently cooled down completely to β_0. By performing multiple temperature reversals, the FORC surface that defines the model is identified. Ultimately, the strain is computed by summation over the vertical trapezoidal surfaces Q_k in the Preisach plane.

Rather than cooling down from a temperature α' to the martensite saturation temperature β_0 during the temperature reversals, FORC increments can also be identified by heating from a temperature β' to the austenite saturation temperature α_0. In this case, a FORC increment can intuitively be interpreted as the strain that "enters" the actuator when heated from β' to α', assuming the actuator initially started from saturated austenite. Therefore, a heating FORC increment can be expressed as

$$F(\beta', \alpha', \sigma) = f_{\beta'}(\sigma) - f_{\beta', \alpha'}(\sigma). \tag{3.2}$$

Similarly, the strain output of the Preisach model defined by a heating FORC surface can be defined. While the earlier definition (Equation 2.17) is based on the summation of the vertical trapezoids Q_k, the strain output for the heating model can be calculated based on the horizontal trapezoids in the Preisach plane as

$$\epsilon(t, \sigma) = \begin{cases} \displaystyle\sum_{k=1}^{n(t)-1} \Big[F(\beta_k, \alpha_{k-1}) - F(\beta_k, \alpha_k)\Big] + F(\beta_n, \alpha_{n-1}), & \text{if } \dot{T}(t) < 0 \text{ (cooling)}, \\ \displaystyle\sum_{k=1}^{n(t)} \Big[F(\beta_k, \alpha_{k-1}) - F(\beta_k, \alpha_k)\Big], & \text{if } \dot{T}(t) > 0 \text{ (heating)}. \end{cases} \tag{3.3}$$

The dual Preisach model, containing both heating and cooling FORC surfaces, therefore outputs two strain values: one based on the cooling data and one based on the heating data. During operation, the corresponding strain value—depending on whether the actuator is heating or cooling—is used to estimate the actuator strain. It should be noted that a gap is expected during changes in the temperature direction. The dual Preisach model strain, including both heating and cooling FORCs, can be expressed as

$$\epsilon(t, \sigma) = \begin{cases} \displaystyle\sum_{k=1}^{n(t)} \Big[F_{\text{cooling}}(\alpha_k, \beta_{k-1}) - F_{\text{cooling}}(\alpha_k, \beta_k)\Big], & \text{if } \dot{T}(t) < 0 \text{ (cooling)}, \\ \displaystyle\sum_{k=1}^{n(t)} \Big[F_{\text{heating}}(\beta_k, \alpha_{k-1}) - F_{\text{heating}}(\beta_k, \alpha_k)\Big], & \text{if } \dot{T}(t) > 0 \text{ (heating)}. \end{cases} \tag{3.4}$$

Moreover, the method used to measure the FORC increments was further improved. During experimentation, it was observed that drift in the thermocouple signal occurred only during heating and was therefore likely caused by interference from the PWM module. Consequently, a new and arguably simpler method can be used to identify FORC increments. Previously, during the identification process for a cooling FORC surface, the temperature was increased from the martensite saturation temperature to a temperature α' and subsequently cooled down in controlled steps to multiple intermediate temperatures β' until reaching β_0. A time interval of 45 s was maintained between the steps to ensure that the set duty cycle reached a steady-state temperature.

This method can be simplified by following the same approach, but instead of reversing from α' in discrete steps, the actuator is cooled down immediately to β_0 while simultaneously sampling the strain and thermocouple readings. Similar to the PWM-based identification approach, the Preisach grid is discretized into equidistant PWM grid points. As before, the FORC surface identification consists of multiple experiments. In each experiment:

- The actuator is heated to α'. Note that this represents a duty cycle rather than a temperature.

- The actuator is then cooled down completely to the martensite saturation temperature β_0, while sampling the strain and temperature at 2 Hz.

Following this approach, the FORC increment of each sample can be evaluated, and a polynomial fit can be applied as shown in Figure 3.2. Since α' is set as a PWM duty cycle (rather than a temperature), the corresponding temperature is estimated based on the first measurement of the reversal curve. Using this fit, the value of each FORC increment along the reversal curve can be identified.

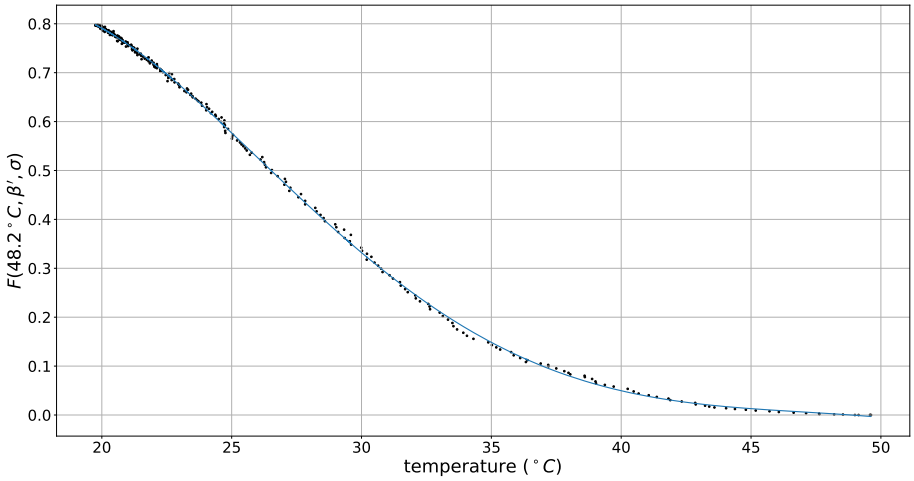


Figure 3.2: Polynomial fit of 5th degree (blue) applied to the temperature-FORC data (black) from a temperature reversal. In this experiment, the PWM duty cycle was increased to approximately 13.2 %, corresponding to an α' of 48.2 °C, and then cooled down to β_0.

Similarly, the FORC increments for the heating surface can be identified by sampling the strain and temperature measurements during heating and estimating the FORC increments using a polynomial fit. It should be noted, however, that this identification method will likely yield less accurate results than the cooling identification, due to the presence of thermocouple drift during heating.

3.3. Preisach Model Results and Evaluation

3.3.1. Model Results

The major hysteresis loops of the temperature–-strain relationship of the Preisach models calibrated using cooling FORC increments and heating FORC increments are provided in Figure 3.3b and Figure 3.4b, respectively. In each figure, the red lower branch shows the model behavior during increasing temperature, while the red upper branch shows the model behavior during decreasing temperature. The measured strain data (black) was obtained by heating the actuator from approximately 20 °C to approximately 65 °C and subsequently cooling it back to approximately 20 °C. In each figure, the left plot corresponds to the load case of 2.5 N, while the right plot corresponds to the load case of 3.0 N. Unlike the previous identification procedure, the load cases for 1.5 N and 2.0 N were not calibrated due to the project's time constraints. A simplified version of the Preisach update function used to compute the strain is provided as pseudocode in Appendix A.

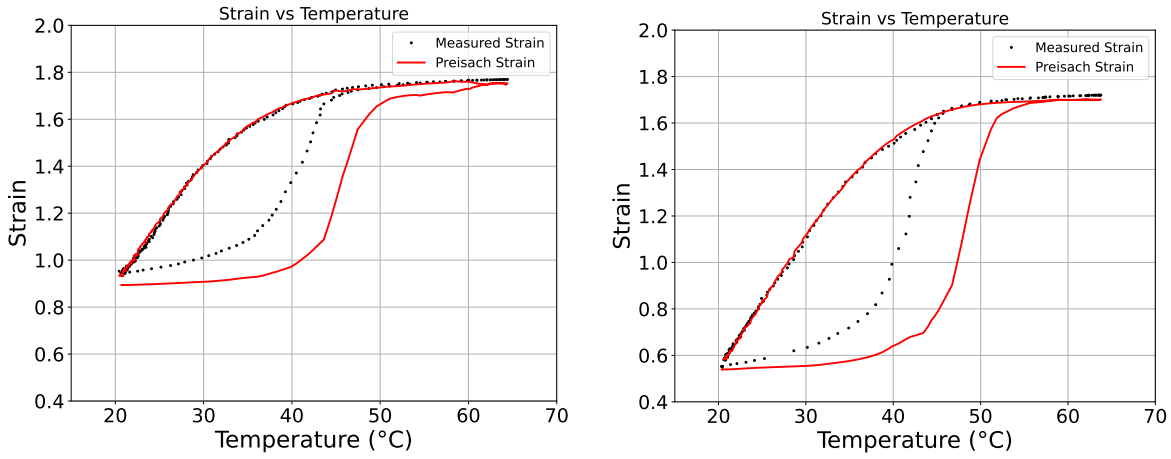


Figure 3.3: Temperature–strain major hysteresis loops of the Preisach model identified using the simplified FORC identification with polynomial curve fitting for cooling FORCs. The figures compare the identified model (red) with the experimental data (black). (a) Temperature–strain response at 2.5 N and (b) temperature–strain response at 3.0 N.

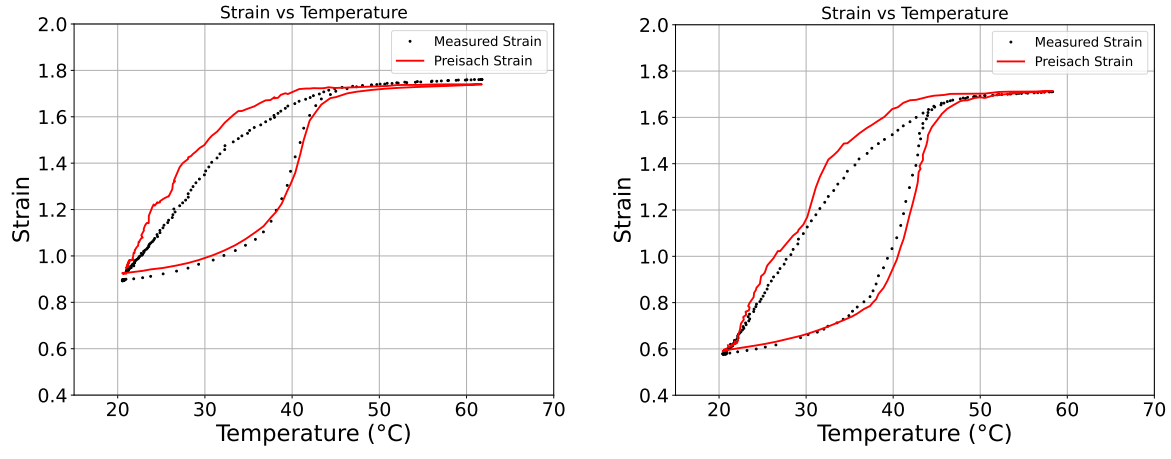


Figure 3.4: Temperature–strain major hysteresis loops of the Preisach model identified using the simplified FORC identification with polynomial curve fitting for heating FORCs. The figures compare the identified model (red) with the experimental data (black). (a) Temperature–strain response at 2.5 N and (b) temperature–strain response at 3.0 N.

Similarly, Figure 3.5 shows the dual Preisach model compared to the experimental strain data, where the model output is computed using two FORC surfaces: one for cooling and one for heating, as described in Equation 3.4.

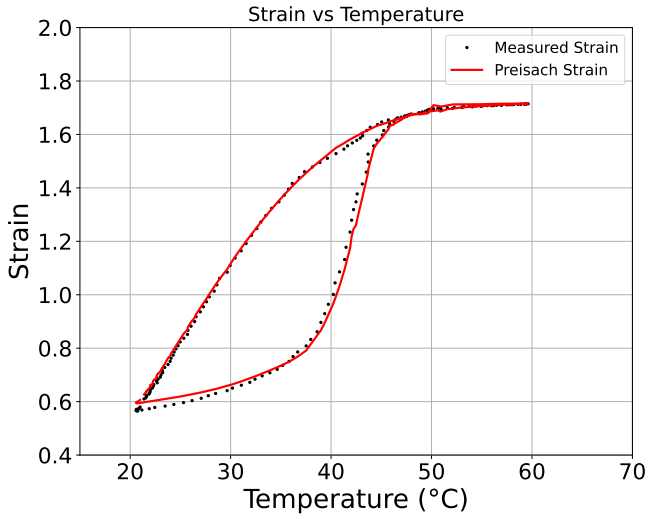
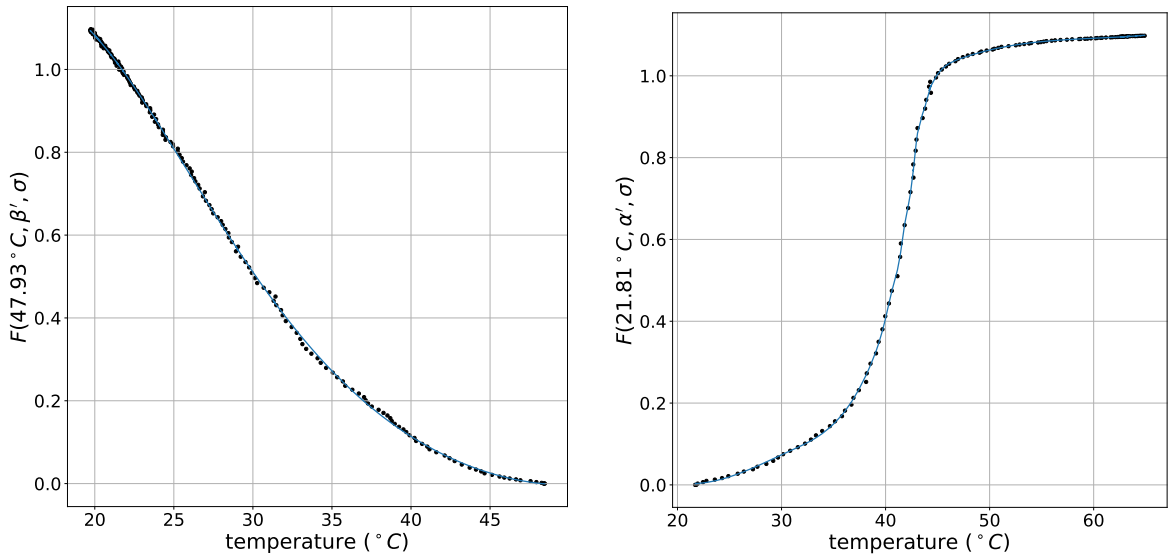


Figure 3.5: Temperature–strain major hysteresis loop of the combined model using both the heating and cooling FORC surfaces for a load of 3.0 N. The model is shown in red and the experimental data in black.

3.3.2. Model Evaluation

Evaluation of the Identification Method

As explained in subsection 3.2.2, the identification process was simplified by fitting a curve through the temperature–FORC data, rather than setting a prescribed PWM value and estimating the steady-state temperature based on the actuator length. An example of a cooling FORC experiment is provided in Figure 3.6a, while an example of a heating FORC experiment is provided in Figure 3.6b.



(a) **(b)**

Figure 3.6: Two examples of 5th degree polynomial fits (blue) applied to the temperature–FORC data (black) from temperature reversals obtained during post-processing of the dual Preisach model. (a) Polynomial fit of a cooling FORC experiment, where the temperature is increased to α' (47.93 °C) from martensite saturation and subsequently cooled down to approximately 20 °C. (b) Polynomial fit of a heating FORC experiment, where the temperature is decreased to β' (21.81 °C) from austenite saturation and subsequently heated up to approximately 65 °C.

Compared to the identified Preisach models described in the previous chapter, this calibration method results in smoother temperature–strain curves, especially for the models identified during cooling. The fitted curve produces smooth transitions between FORC increments and therefore reduces oscillatory behavior in the model. The main source of error arises from estimating the reversal temperature: α' in the case of cooling identification and β' in the case of heating identification. As discussed earlier, this temperature is based on the temperature reading of the first measurement of the reversal curve.

For the cooling identification process, no drift is present in the temperature readings since the PWM module does not actuate the SMA during cooling. For the heating identification process, however, the drift in the temperature readings causes a larger error in the estimated β'. This explains the more irregular behavior of the temperature–strain loops in Figure 3.4 compared to Figure 3.3.

Therefore, the heating model could be improved by performing temperature measurements only when the PWM signal is temporarily switched off. By briefly switching off the heating, waiting a millisecond (or even less), performing the measurement, and immediately turning the heating back on, the actuator temperature could be estimated without drift. If this process is performed sufficiently quickly, the short interruption in heating will have a negligible effect on the identification process.

Evaluation of the Model

The results from the newly identified models confirm that the poor behavior discussed in Appendix D is linked to the identification procedure. Previously, it was shown that the model identified using FORC increments measured during cooling performed well only during cooling. As Figure 3.4 shows, the model identified using a FORC surface measured during heating shows good agreement with the experimental data during heating only.

This indicates that the kinematic behavior of nitinol during the cooling solid-state transition differs from the kinematic behavior during the solid-state heating transition. By calibrating the Preisach model using data from only cooling or only heating, the model cannot capture the behavior of the opposite transition. Mathematically, this behavior can be explained by considering the Preisach density function $\mu(\alpha, \beta)$ and the Preisach strain equation from section 2.3, which is repeated below for clarity:

$$\epsilon(t) = \iint_{\alpha \geq \beta} \mu(\alpha, \beta)\, \gamma_{\alpha,\beta}[T](t)\, \mathrm{d}\alpha\, \mathrm{d}\beta. \tag{3.5}$$

As previously discussed, the identification procedure using FORC increments provides a method for estimating $\mu(\alpha, \beta)$. Because the classical Preisach model assumes that $\mu(\alpha, \beta)$ is a static material property, while the experimental results indicate that accurate strain estimates are only obtained when $\mu(\alpha, \beta)$ is direction-dependent, this explains the poor performance of a model calibrated exclusively during either heating or cooling.

Therefore, to accurately model the behavior of the actuator, the density function must be calibrated separately for heating and cooling. This is done in the dual Preisach model, which uses two FORC surfaces—one for heating and one for cooling—and therefore estimates two different density functions. The results of this two-way model, shown in Figure 3.5, show that this approach significantly improves the model's strain estimates for the major hysteresis loop.

The downside of this approach is that a discontinuity appears when switching between the FORC surfaces. An example is shown in Figure 3.7, where the model is heated from a low temperature and subsequently cooled down. At the temperature reversal—that is, when switching from the heating model to the cooling model—the predicted strain shows a sudden drop to a low value. This behavior results from the behavior of the cooling model, which, as discussed earlier, cannot accurately capture the heating behavior of the SMA. During heating from a low temperature to approximately 40°C (see Figure 3.7), the cooling model predicts a strain that is too low. When the temperature is then reversed, this incorrect strain value is used as the reversal reference, which leads to a poor prediction of the subsequent minor hysteresis loop.

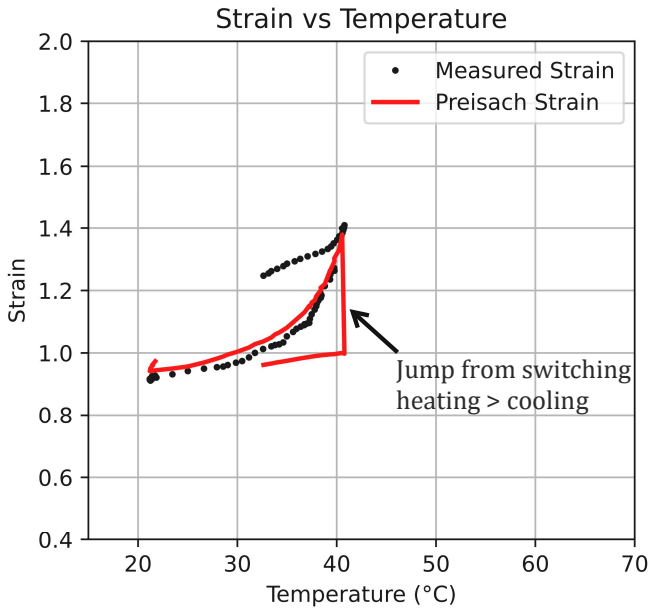
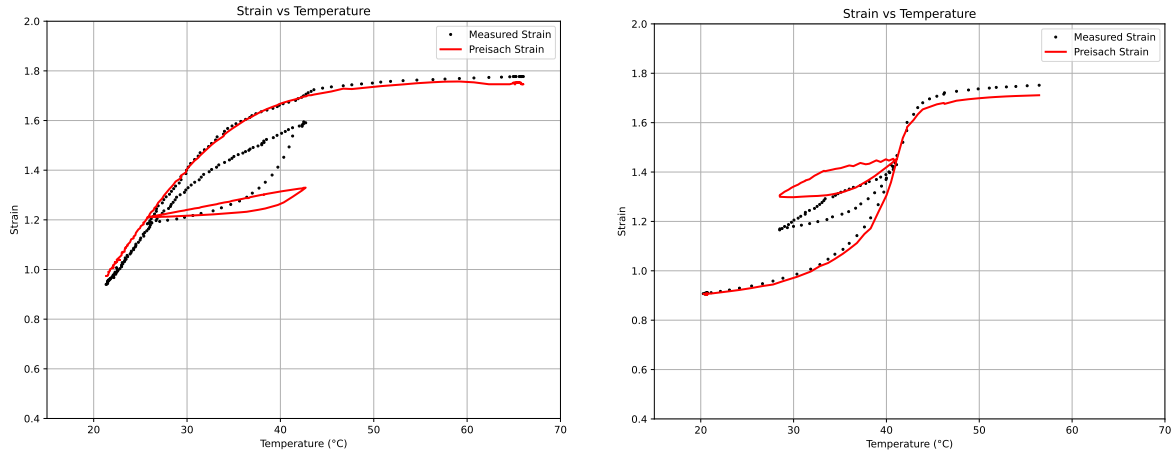


Figure 3.7: Temperature–strain relationship of the model (red) and the data (black) under a constant load of 2.5 N. The actuator is heated from approximately 20°C to 40°C and subsequently cooled down to approximately 32°C. At the temperature reversal, a noticeable jump is visible in the model, caused by the difference between the heating and cooling data used in the model.

In other words, for minor hysteresis loops to be predicted accurately, the model must capture both the heating and cooling behavior. Two more examples illustrating this issue are shown in Figure 3.8. Figure 3.8a shows part of the major hysteresis loop during cooling, including a minor loop where the model switches to heating FORCs and then back to cooling FORCs. Figure 3.8b shows part of the major hysteresis loop during heating, including a minor loop where the model switches to cooling FORCs and then back to heating FORCs. These results show that, while the model agrees well with the experimental data along the major hysteresis loop, it performs poorly when predicting minor hysteresis loops.



(a) (b)

Figure 3.8: Temperature–strain relationship of the dual-Preisach model (red) and experimental data (black). The model shows good agreement with the experimental data along the major hysteresis loop, but poor agreement for the minor hysteresis loops. (a) Upper branch of the major hysteresis loop, including a minor loop. The temperature is decreased from approximately 65°C to 26°C, increased to approximately 42°C, and finally decreased to approximately 22°C. (b) Lower branch of the major hysteresis loop, including a minor loop. The temperature is increased from approximately 20°C to 41°C, decreased to approximately 28°C, and finally increased to approximately 56°C.

To conclude, the model shows very good agreement with the experimental data for the major hysteresis loop. By capturing the model behavior using separate FORC surfaces—one for heating and one for cooling—the Preisach density function is effectively represented by two different density functions. The downside of this approach is that the model still cannot accurately capture the behavior of minor hysteresis loops, since these rely on a density function that is consistent for both heating and cooling.

This suggests that the calibration method based on FORC increments may not be a good approach for accurately capturing the behavior of SMAs. Therefore, an alternative calibration method could be considered to improve the model. One such method is described in [34], where the Preisach density is identified using a least-squares optimization algorithm. The method consists of performing a series of excitation experiments. During each experiment, the state of every hysteron (i.e., whether it is switched on or off) is tracked, while the corresponding strain is measured. Repeating this process for different input trajectories results in a dataset that relates the hysteron states to the measured strain. Based on this dataset, an optimization algorithm can be solved to determine the set of weights that best fit the SMA behavior, thereby approximating the Preisach density function. Since this calibration approach uses both heating and cooling paths, it could provide a more accurate representation of SMA behavior than the FORC increment method.

3.4. Baseline Feedback Control Using a PID Controller

3.4.1. PID Controller Design and Implementation

A block diagram of the implemented PID controller is shown in Figure 3.9. The controller computes the strain error e as the difference between the reference strain ϵ_r and the measured strain ϵ_m, obtained from the encoder. The controller output $u(t)$ is the signal sent to the PWM module and is expressed as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}, \tag{3.6}$$

where K_p, K_i, and K_d are the tuning gains corresponding to the proportional, integral, and derivative terms, respectively.

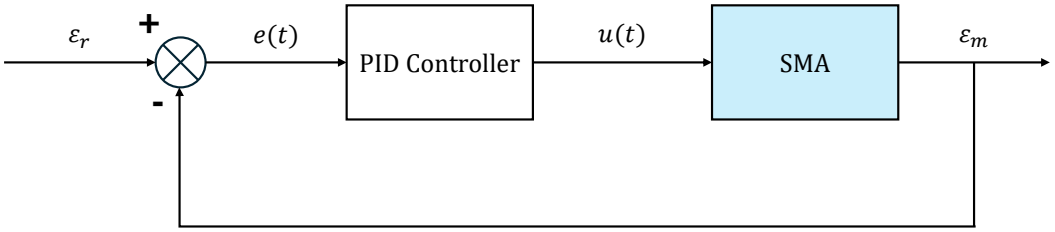


Figure 3.9: Control scheme of the PID controller. The reference strain is denoted by ϵ_r, the strain error by $e(t)$, the controller output $u(t)$ is expressed as a PWM duty cycle, and ϵ_m represents the strain measured by the encoder.

Note that $u(t)$ represents the PWM duty cycle, as it is the signal sent to the PWM module. Consequently, the controller output is bounded and saturates within the interval $[0, 1]$, corresponding to a duty cycle range of 0% to 100%.

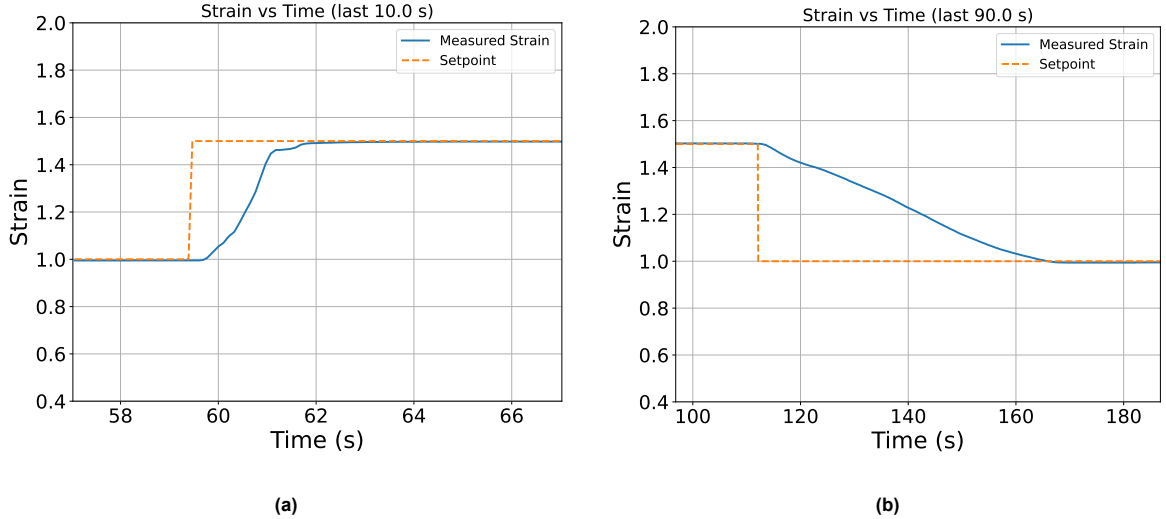
As stated earlier, no non-linear gain compensation is implemented in this control scheme. Although such compensation could improve the tracking performance due to the non-linear temperature–strain response of SMAs, it is omitted since the PID controller is intended to serve as a baseline controller. A limitation of this approach is that the controller behavior is expected to vary across different temperature ranges. In regions where the SMA strain is highly sensitive to temperature fluctuations, the response will differ from regions where the strain is relatively insensitive to temperature changes. The main advantage of this controller is its simplicity, as it only requires a strain estimate. In addition, the output $u(t)$ can be applied directly by adjusting the PWM duty cycle, without introducing noticeable delays.

The PID controller is implemented in a control loop in Python running at 40 Hz. As explained in section 2.4, the microcontroller sends the measurements—in this case the encoder position—to Python via serial communication, where the strain and strain error are computed and the PID loop is updated. Subsequently, Python sends a PWM command back to the microcontroller. The controller gains were tuned experimentally by first applying only a proportional gain, then increasing the integral gain, and finally damping the response by increasing the derivative gain. Since the encoder provides incremental data, a filter could be implemented to prevent chattering in the control output caused by the derivative term. However, no such chattering was observed during real-time implementation, and therefore no filter is applied to the measured encoder data.

Since the limited bandwidth of SMA actuators is primarily due to slow cooling times, the controller gains were tuned to prevent overshoot of the reference strain. While undershoot can be rapidly corrected by heating the actuator, even a small overshoot requires a long time to recover because of the slow cooling. Additionally, the controller gains were selected for the temperature range in which the strain is most sensitive to temperature variations.

3.4.2. Experimental Tracking Performance

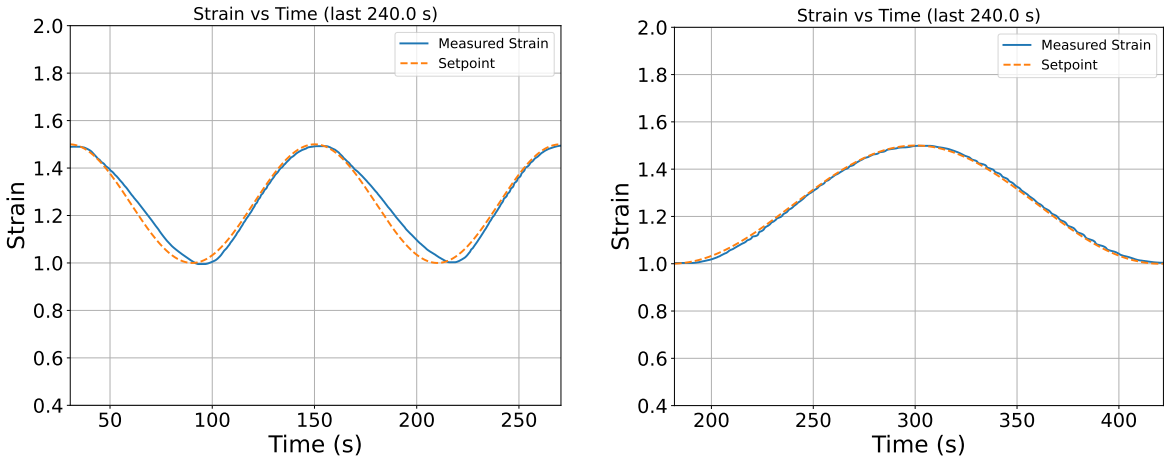
A step-up and step-down response of the PID controller are shown in Figure 3.10a and Figure 3.10b, respectively. The blue curve represents the measured strain obtained from the encoder data, while the orange dashed curve denotes the strain setpoint. In Figure 3.10a, the strain is increased from 1.0 to 1.5, whereas in Figure 3.10b it is decreased from 1.5 to 1.0. As stated earlier, the controllers evaluated in this chapter were only tuned and tested for a fixed load of 2.5 N. Defining the settling time as the duration required for the actuator to reach and remain within $\pm 2\%$ of its setpoint, the step-up response exhibits a settling time of approximately 2 seconds. In contrast, the step-down response is limited by the slow cooling of the actuator, resulting in a significantly longer settling time of approximately 50 seconds.



(a) (b)

Figure 3.10: Time–strain plots of the SMA actuator under a 2.5 N load. The blue curve represents the measured strain, while the dashed orange curve indicates the reference strain setpoint. (a) Response to a step increase in strain from 1.0 to 1.5. (b) Response to a step decrease in strain from 1.5 to 1.0.

Furthermore, the tracking performance was evaluated for two slow-moving sinusoidal setpoints: one at $\frac{1}{120}$ Hz, as shown in Figure 3.11a, and one at $\frac{1}{240}$ Hz, as shown in Figure 3.11b. Both cases demonstrate good tracking performance of the PID controller. The error observed during decreases in strain–as visible in Figure 3.11a–is mainly caused by the slow cooling of the actuator. This error is therefore not due to limitations of the controller, but rather to the inherent limitations of the plant.



(a) (b)

Figure 3.11: Time–strain plots of the SMA actuator under a 2.5 N load. The blue curve represents the measured strain, while the dashed orange curve corresponds to a slowly varying sinusoidal reference strain. (a) Response to a sinusoidal reference at $\frac{1}{120}$ Hz. (b) Response to a sinusoidal reference at $\frac{1}{240}$ Hz.

Overall, the tracking performance of the PID controller shows surprisingly accurate results. Upon reaching the setpoint, the output stabilizes without showing chattering. Furthermore, the controller gains were easy to tune, and no significant oscillations are observed upon reaching the setpoint.

As stated in the literature, a limitation of the PID controller is that it cannot capture the non-linear and hysteretic behavior of the actuator. This can be confirmed by the incremental step-up response

shown in Figure 3.12. For the first step input–where the strain setpoint is increased from 0.9 to 1.0–the response of the system is significantly slower than for the last step input–where the strain setpoint is increased from 1.4 to 1.5. This is a result of the highly non-linear temperature–strain behavior of SMA actuators. During heating, the actuator used in this test exhibits high sensitivity in the strain range between 1.2 and 1.6, meaning that a small increase in the PID output $u(t)$ results in a large increase in strain. In contrast, at lower strain ranges, a relatively large control input is required to obtain a similar strain response. This difference in sensitivity explains the variation in settling time observed in Figure 3.12.

As stated before, some form of non-linear gain scheduling could therefore be implemented to improve the tracking performance. However, the temperature–strain sensitivity of SMAs does not always occur within the same temperature or strain range, but depends on the hysteretic state. As a result, the non-linear gain would depend on the hysteretic state of the SMA and would therefore require some type of hysteresis model.

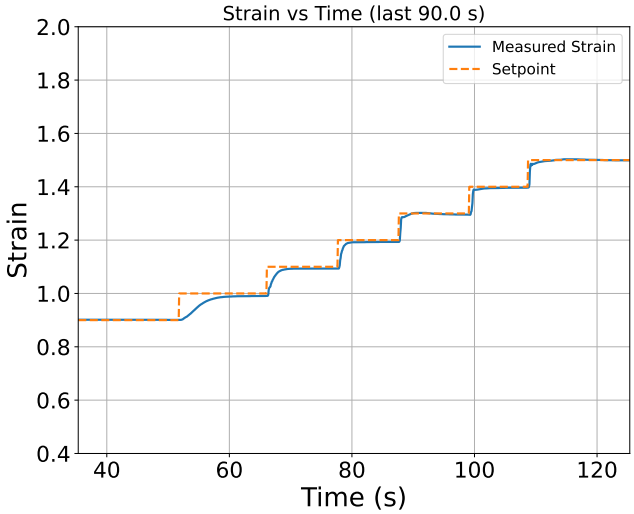


Figure 3.12: Time–strain plots of the SMA actuator under a 2.5 N load. The blue curve represents the measured strain, while the dashed orange curve denotes an incrementally increasing step input strain setpoint.

To conclude, the PID controller is easy to implement, provides good tracking performance without oscillations, and does not exhibit output chattering. However, it is limited by the non-linear behavior of SMAs, resulting in variations in settling time across different strain ranges. This could be addressed by implementing non-linear gain scheduling; however, such an approach is complicated by the hysteretic nature of the temperature–strain response.

3.5. Hybrid Feedforward-Feedback Control

3.5.1. Controller Design and Implementation

To address the aforementioned issue of the PID controller, the dual-Preisach model described in section 3.3 is used in a hybrid control architecture. Similar to methods described in the literature—as explained in section 2.2—this architecture consists of an inverse Preisach model that outputs a feedforward temperature and a PID controller that outputs a feedback temperature, thereby forming a hybrid controller. The feedforward and feedback terms are summed and applied to the PI temperature controller described in Appendix C, resulting in a cascaded control structure. Intuitively, the feedforward term is used to ”push” the system towards the correct setpoint, and the feedback term is used to correct for model inaccuracies and disturbances.

Note that the PID controller in this architecture differs from the one described in the previous section: here it outputs a reference temperature, whereas the previous controller outputs a duty cycle. Fur-

thermore, since the Preisach model described in the previous chapter outputs strain as a function of reference temperature and reference stress, it must be inverted to obtain temperature as a function of strain. This inversion is performed numerically using the bisection method, as described in subsection 3.5.2. The complete source code is provided in Appendix B.

The schematic representation of the hybrid controller is provided in Figure 3.13. The feedforward temperature T_{ff} from the inverted model and the feedback temperature T_{fb} from the PID controller are added to form the setpoint temperature T_{sp}. The measured temperature from the thermocouple, T_m, is subtracted from this setpoint to obtain the temperature error $e_T(t)$, which is used by the temperature controller to track T_{sp} by outputting a PWM duty cycle $u(t)$. Furthermore, the inverse Preisach model uses T_m, the reference stress σ_r, and the reference strain ϵ_r to compute the feedforward term. The measured strain ϵ_m, obtained from the encoder, is used to compute the error signal $e_\epsilon(t)$, which is used by the PID controller to generate the feedback term. Moreover, unlike what the schematic suggests, note that the inverse Preisach model uses the entire temperature history rather than just the current temperature reading T_m, since the Preisach state depends on the entire temperature history.

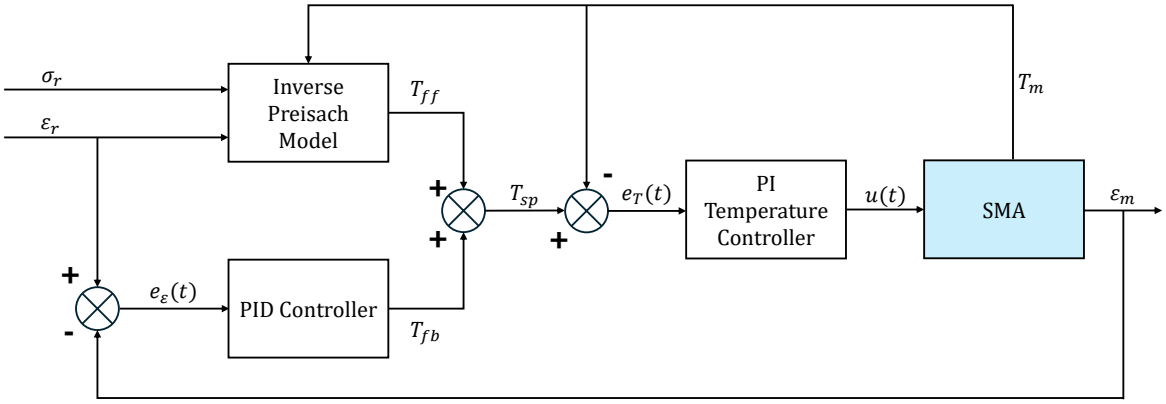


Figure 3.13: Control scheme of the combined feedforward–feedback architecture. The reference strain is denoted by ϵ_r, the reference stress by σ_r, and the strain error by $e(t)$. The inverse Preisach model generates the feedforward temperature T_{ff} corresponding to the desired reference stress and is updated only upon changes in the reference. The PID controller computes the feedback temperature T_{fb} based on $e(t)$ and is activated once the temperature tracking error is within $\pm 1°$C of T_{ff}. The temperature T_m measured by the thermocouple is subtracted from the reference temperature T_{sp} and fed as the error signal $e_T(t)$ to the PI controller, which outputs the control signal $u(t)$ as a PWM duty cycle in the range $[0, 1]$. The measured temperature is fed into the inverse Preisach model, while the measured strain ϵ_m (from the encoder) is used to compute $e_\epsilon(t)$.

During real-time operation, one approach is to continuously update the feedforward temperature. However, experimental results showed that this leads to a highly chattery feedforward signal and, consequently, a chattery system response. This can be explained by the following example: assume the actuator temperature is 20°C and the strain is 0, while the desired strain is 1.0. The true temperature required to reach this strain is 40°C, but the model predicts it to be 38°C. The following sequence then occurs:

- The controller starts heating the actuator to 38°C.
- The PID controller observes that the desired strain is not reached (due to model error) and continues increasing the temperature towards 40°C.
- Since the temperature exceeds the predicted 38°C, the model predicts that the temperature should be reduced. Due to hysteresis, it now outputs a much lower temperature (e.g., 25°C).

As a result, the PID controller drives the temperature upward (to reduce strain error), while the model simultaneously commands a significantly lower temperature. This conflict leads to oscillatory behavior and poor tracking performance.

To avoid this issue, a different strategy is implemented: the inverse Preisach model updates the feedforward term only when the strain setpoint ϵ_r changes. Furthermore, to prevent undesired integrator wind-up, the PID controller is only activated when the setpoint temperature is within $\pm 1^{\circ}$C of the feedforward temperature. Outside this range, the PID controller remains idle. Without this mechanism, the integral term would accumulate a large error while waiting for the temperature to approach the setpoint, resulting in overshoot.

With this control strategy, the feedforward term effectively replaces part of the integral action of the PID controller. Instead of relying only on feedback—where the integral term must converge to the steady-state value within the hysteresis loop—the required control effort is partially predicted a priori by the feedforward component. Therefore, this strategy theoretically allows the controller to achieve more consistent settling behavior across different strain ranges, improving upon the performance observed in the previous section with a pure PID controller.

The main drawback of this controller is its increased complexity and the number of required state estimates. Unlike the PID controller described in the previous section, which only requires a strain measurement, the hybrid controller also relies on temperature and stress estimates. These additional states complicate the design of a robotic tentacle, as they require extra sensors or state-prediction methods. Moreover, the thermocouple has a limited bandwidth of less than 10 Hz, which makes fast tracking challenging. As a result, the cascaded controller structure is likely to show slower response times compared to the pure PID controller.

3.5.2. Inverse Preisach Model for Feedforward Compensation

Since the dual-Preisach model described in section 3.3 outputs strain rather than temperature, it must be inverted for control purposes. The method described in [14] achieves this by rearranging the numerical data within the FORC surfaces to construct inverse FORC surfaces in the form of 3D lookup tables. While this approach is computationally efficient, it introduces implementation complexity. Therefore, to simplify the process, a bisection search algorithm is used to compute the inverse temperature in real-time.

During real-time operation, the Preisach model—as described in Appendix A—is continuously updated within the control loop, since its state depends not only on the current temperature but also on the full temperature history. When the setpoint changes, the control loop freezes the current Preisach state and searches for the temperature that, starting from this state, results in the desired strain setpoint. The bisection method achieves this by repeatedly evaluating the Preisach function at different temperatures until a value is found that produces a strain within a specified tolerance of the desired strain. A more detailed description of the bisection method is provided in [10], and the corresponding source code is included in Appendix B.

3.5.3. Experimental Tracking Performance

As stated in section 3.3, the model shows good agreement with experimental data for the major hysteresis loop, but poor agreement for the minor loops. Therefore, good tracking performance can be expected when starting from a saturated martensite or austenite state, while poor performance is expected from intermediate martensite–austenite states. To first evaluate the controller behavior on the major loop, Figure 3.14 shows the step response when heating from (cold) saturated martensite in Figure 3.14a, and the step-down response when cooling from (hot) saturated austenite in Figure 3.14b.

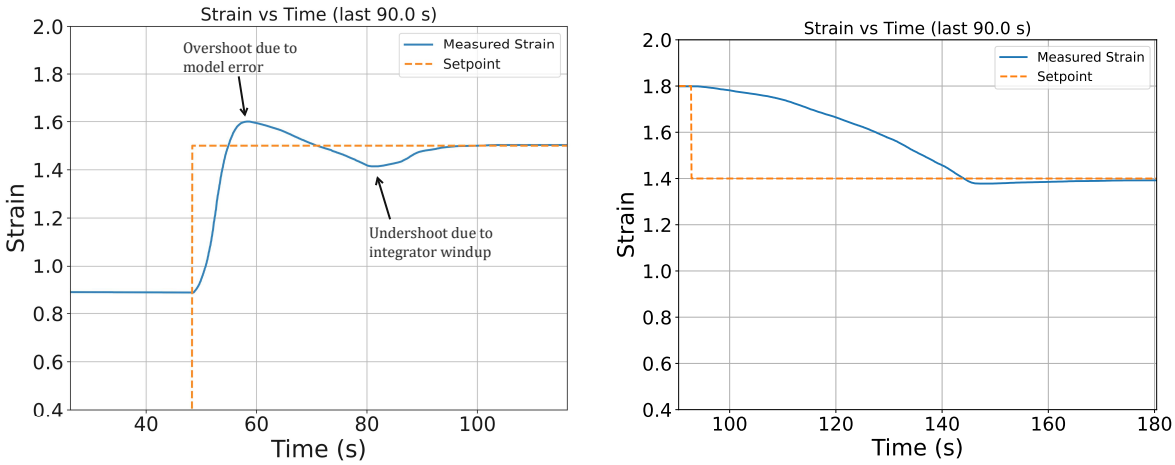


Figure 3.14: Time–strain step responses of the actuator under a fixed load of 2.5 N. The orange curve denotes the strain setpoint, and the blue curve the measured strain. (a) Response to a step-up input from saturated martensite, showing overshoot due to model error and undershoot due to integrator windup. (b) Response to a step-down input from saturated austenite.

Although good performance was expected, the step-up response shows a large overshoot. Further experiments revealed that this overshoot mainly occurs for large step inputs, such as in Figure 3.14a, while it decreases for smaller step inputs. It was found that the actuator behavior depends on the rate of strain change: large heat inputs lead to fast strain changes, which affect the temperature–strain relationship. The temperature–strain behavior of the model and measurements for slow heating is shown in Figure 3.15a, and for fast heating in Figure 3.15b. The slow heating case shows good agreement with the model, as the model was calibrated under equal conditions. For fast heating, however, the model underestimates the strain, which explains the overshoot observed in Figure 3.14a.

This difference in response may be related to system inertia, as the actuator is attached to a mass that builds up velocity with increasing strain rate. Since the strain rate for a given step input depends on the applied heat input—and thus on the gains of the temperature controller—the overshoot could be reduced by lowering these gains. However, doing so would introduce additional lag and degrade overall performance. Furthermore, the undershoot observed in the step-up response is caused by integrator wind-up in both the PID controller and the PI temperature controller. The slow cooling dynamics allow the integral terms to accumulate a large negative value. The performance of the overall system could therefore be improved by retuning the saturation limits of the integral terms.

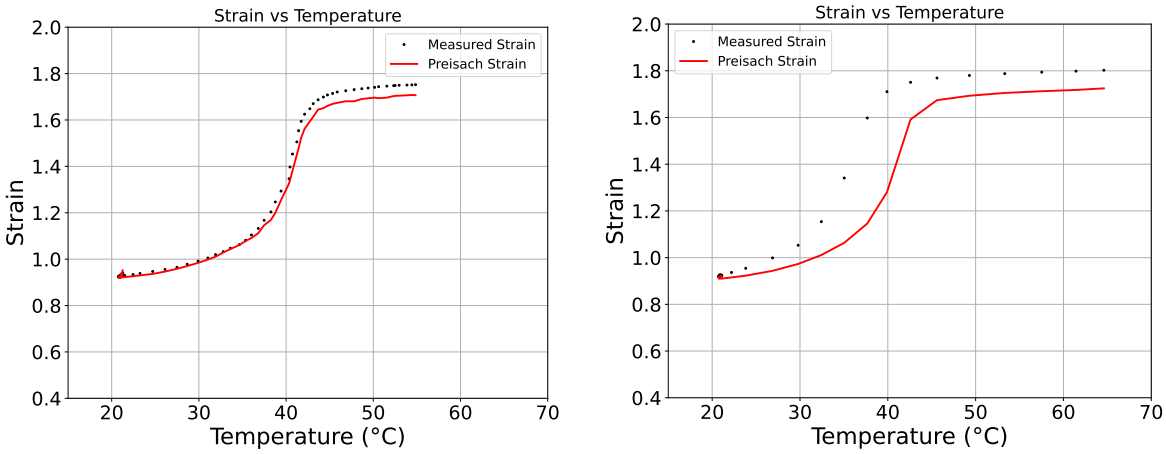


Figure 3.15: Temperature–strain responses of the model (red) and measured strain (black) for different heating inputs. (a) Response when the actuator is heated with 0.5 A. (b) Response when the actuator is heated with 3.0 A.

For smaller step inputs, the overshoot caused by differences in actuator response due to varying strain rates disappears, as shown by the incremental step response in Figure 3.16. As described in subsection 3.5.1, the controller updates the feedforward term upon a change in setpoint. Once the measured temperature is within the specified threshold of the feedforward temperature, the PID controller exits its idle state and begins outputting a feedback temperature to compensate for model error.

This behavior is clearly visible in the first step of the incremental response: at a strain of approximately 1.18, the feedforward temperature falls within the threshold of the measured temperature, activating the PID controller. At this point, a noticeable “bump” appears in the response, corresponding to the addition of the PID control effort. Compared to the PID controller described in section 3.4, this controller shows relatively slow settling times for small setpoint changes (> 10 s).

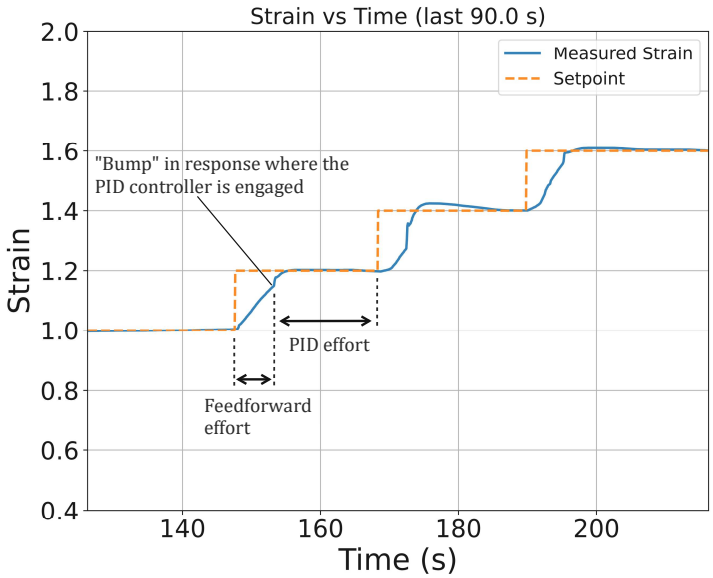


Figure 3.16: Time–strain incremental step response of the actuator under a fixed load of 2.5 N. The orange curve denotes the strain setpoint, and the blue curve the measured strain. A distinct bump in the response corresponds to the addition of the PID control effort.

As stated before, the model performs well for the major hysteresis loop, but performs poorly for minor loops. In these cases, the model prediction—and therefore the feedforward temperature—can deviate significantly from the true value. Two examples illustrating this behavior are shown in Figure 3.17.

In Figure 3.17a, the actuator is heated from (cold) saturated martensite to an intermediate temperature, resulting in a strain of 1.4. During this trajectory, an overshoot occurs at approximately $t = 20$ s, after which the controller cools the actuator back to the setpoint. This reversal introduces a minor hysteresis loop in the model. As a result, for the setpoint of 1.2, the model outputs a feedforward temperature that is too high, causing the controller to initially increase the strain instead of decreasing it. A similar effect is observed in Figure 3.17b, where overshoots at the third and fourth step inputs create minor loops in the model, leading to incorrect feedforward actions. Therefore, non-monotonic heating or cooling can introduce large model errors and degrade controller performance.

This issue could be mitigated by adding a conditional update rule in the controller: if the strain setpoint is higher than the current strain while the feedforward temperature is lower than the current temperature, the feedforward term should not be updated. Similarly, if the setpoint is lower than the current strain while the feedforward temperature is higher than the current temperature, the feedforward term should remain constant. However, due to time constraints, such a method was not implemented in this research.

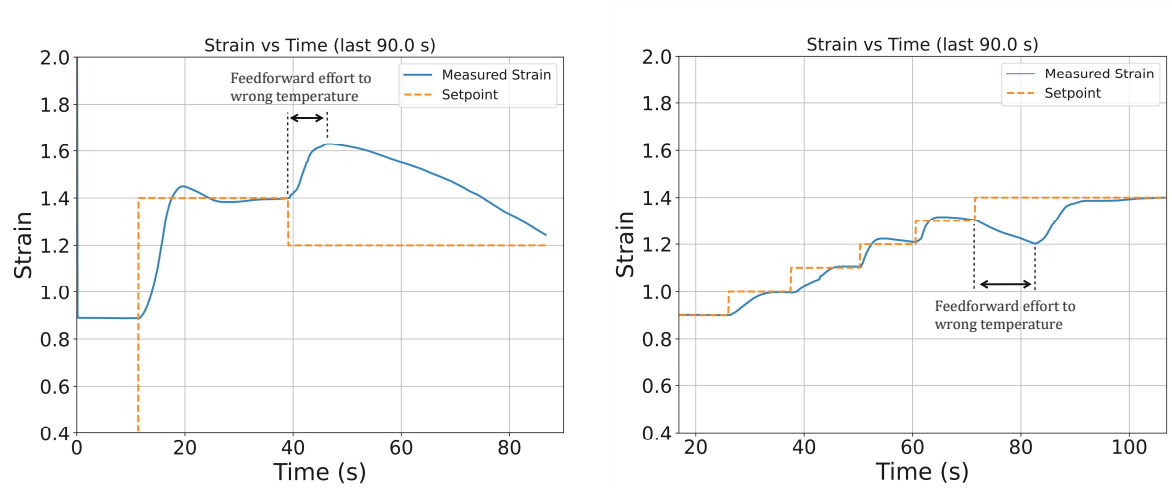


Figure 3.17: Two examples of time–strain step responses of the actuator under a fixed load of 2.5 N, showing feedforward temperature predictions based on minor hysteresis loops. The orange curve denotes the strain setpoint, and the blue curve the measured strain. (a) Example where the model predicts a feedforward temperature that is too high. (b) Example where the model predicts a feedforward temperature that is too low.

Since the feedforward term is updated upon setpoint changes, the controller's tracking performance was evaluated for a continuously varying sinusoidal setpoint of $\frac{1}{240}$ Hz. However, the tracking performance is limited due to conflicts between the feedforward and feedback terms. In the example shown in Figure 3.18, the feedforward term drives the actuator towards a lower temperature (due to model error), while the feedback term increases the temperature to reduce the strain error, resulting in oscillatory behavior. This behavior is expected, as the feedforward term is designed to bring the system close to the desired setpoint rather than correct small tracking errors. The tracking performance observed in Figure 3.18 could therefore be improved by relying solely on the feedback term. However, since the objective of this section is to evaluate the model performance, this modification was not implemented.

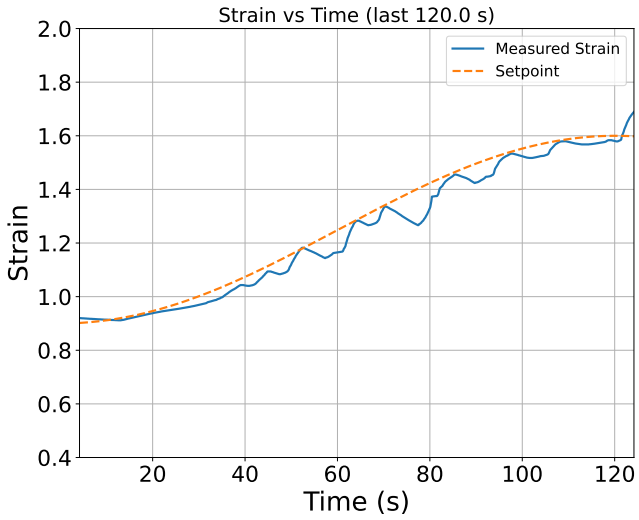


Figure 3.18: Time–strain response of the actuator under a fixed load of 2.5 N. The orange curve denotes a sinusoidal strain setpoint of $\frac{1}{240}$ Hz, and the blue curve the measured strain.

Overall, the performance of the feedforward–feedback controller is worse than that of the pure PID controller; it exhibits poorer tracking performance, slower settling times, and larger errors due to differences between the actuator response and the response predicted by the dual-Preisach model. It should be noted, however, that the slow settling times could be improved by using a higher-quality thermocouple. The currently installed thermocouple has a low update rate, which limits the settling time of the temperature controller. Even with a higher-quality sensor, fast tracking is not guaranteed, as thermal lag between the actuator and the thermocouple tip will still be present.

Furthermore, since the controller performs reliably only when the feedforward term is based on major hysteresis loops, improving the model calibration—such as with the method described in subsection 3.3.2—could reduce errors for minor loops and improve overall control performance. Alternatively, the feedforward action could be applied more conservatively. In the current implementation, the feedforward term is set equal to the temperature predicted by the model, which can lead to over- or undershoot when model errors are present. Reducing the feedforward contribution would make the controller less sensitive to these errors, while still contributing to moving the actuator toward the desired setpoint.

The main advantage of the feedforward–feedback controller is its consistency across different strain ranges. As discussed in subsection 3.4.2, the pure PID controller exhibits varying settling times depending on the strain range. Although the feedforward–feedback controller is slower and less accurate, it achieves more uniform settling behavior across these ranges.

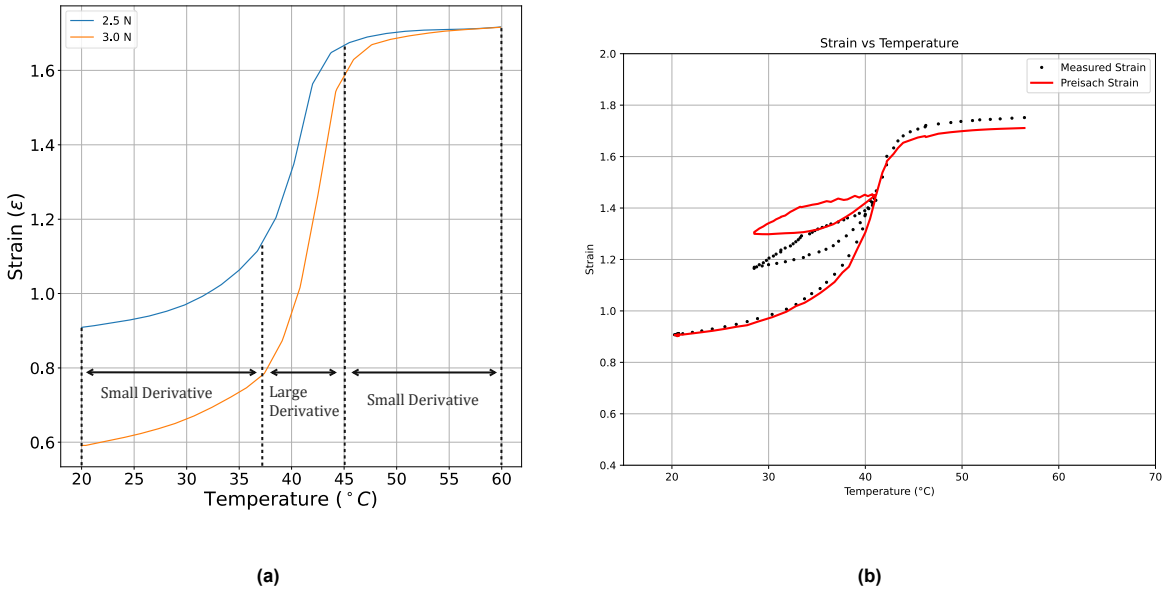
3.6. Future Controller Improvements

As discussed earlier, the PID controller showed very good tracking performance and settling times, but suffered from varying response rates across different strain ranges. This behavior is caused by the nonlinear temperature–strain relationship of the actuator. The model used in the FF-FB controller, on the other hand, exhibits large prediction errors but shows good consistency across strain ranges. Therefore, a new controller could be designed that is similar to the PID controller described in section 3.4, but uses the (dual) Preisach model for gain scheduling, thereby implementing a model-based gain scheduling approach.

More specifically, it was observed that the strain–temperature derivative $\frac{d\epsilon}{dT}$ follows a similar trend across load ranges. The heating branch of the major hysteresis loop for loads of 2.5 N and 3.0 N is shown in Figure 3.19a, illustrating this behavior. Although $\frac{d\epsilon}{dT}$ varies slightly between loads, the differences are limited, with a relatively high derivative between approximately 37°C and 45°C, and a lower derivative outside this range. Similarly, from the heating branch discussed in section 3.3 and

shown again in Figure 3.19b, it can be seen that while the model prediction of strain is inaccurate, the derivative $\frac{d\epsilon}{dT}$ agrees reasonably well with that of the measured data.

Therefore, even though the model poorly predicts strain in minor hysteresis loops, its derivative trend can still be used to implement a gain scheduling strategy in the PID controller. By doing so, the controller could combine the simplicity and strong tracking performance of the PID controller with the consistent behavior across temperature and strain ranges provided by the model.



(a) (b)

Figure 3.19: Temperature–strain relationships illustrating the trend in the strain–temperature derivative $\frac{d\epsilon}{dT}$. (a) Plot showing that the derivative is relatively consistent across different actuator loads. (b) Plot showing that the derivative of the model and measured strain is similar, despite poor agreement in the strain itself.

3.7. Controller Comparison and Selection

To compare the PID and FF-FB controllers and select the most suitable one for the soft robotic tentacle, they are evaluated based on the criteria below. The corresponding performance indicators are provided in Table 3.1.

Reference tracking performance

Evaluates the ability of the controller to track the reference setpoint. The PID controller showed strong performance, whereas the FF-FB controller exhibited poor tracking when minor hysteresis loops caused incorrect model predictions.

Settling time

Evaluates the settling time, defined as the time required for the actuator to reach and remain within $\pm 2\%$ of the setpoint. For step inputs, the PID controller achieved settling times of approximately 2 seconds, while the FF-FB controller required around 10 seconds.

Steady-state error

Evaluates the ability of the controller to remain close to the setpoint without significant oscillations. Both controllers performed well, although the FF-FB controller showed slightly more oscillatory behavior, likely due to thermal lag in the temperature measurements.

Implementation complexity

Evaluates the effort required to implement the controller. The PID controller is straightforward to implement due to its simplicity and widespread use. In contrast, the FF-FB controller requires significantly

more effort due to the complexity of the Preisach model, its software implementation, calibration, and data processing.

Computational requirements

Evaluates the computational effort per control loop iteration. Although the FF-FB controller requires more computations, this is easily performed with modern computers, resulting in a medium rating.

Number of required state estimates

Evaluates the number of state estimates needed to compute the control output. The FF-FB controller requires temperature, strain, and stress estimates, whereas the PID controller only requires strain. Therefore, the PID controller scores higher on this criterion.

Tuning effort

Evaluates the effort required to tune the controller, based on the number of parameters and the time required. The PID controller has three parameters (proportional, integral, and derivative gains). The FF-FB controller includes three gains for the PID controller, two gains for the temperature controller, and a threshold parameter for switching behavior. Consequently, significantly more tuning effort is required for the FF-FB controller.

Flexibility and adaptability

Evaluates how easily the controller can be adapted to a different SMA actuator. The PID controller scores highly due to its simplicity. The FF-FB controller scores lower, as it requires re-calibration of the Preisach model and more extensive tuning.

Response consistency

Evaluates the consistency of transient behavior across different temperature or strain ranges. The PID controller performs poorly in this regard due to its lack of system knowledge, whereas the FF-FB controller demonstrates more consistent behavior.

Criterion	**PID Controller**	**FF-FB Controller**
Reference tracking performance	*Good*	*Poor*
Settling time	*Good*	*Medium*
Steady-state error	*Good*	*Good*
Implementation complexity	*Good*	*Poor*
Computational requirements	*Good*	*Medium*
Number of required state estimates	*Good*	*Poor*
Tuning effort	*Good*	*Poor*
Flexibility and adaptability	*Good*	*Medium*
Response consistency	*Poor*	*Good*

Table 3.1: Qualitative comparison of the evaluated controllers based on performance and implementation criteria.

To summarize the preceding sections, the PID controller outperformed the FF-FB controller in most scenarios. The main advantage of the FF-FB controller is its more consistent response across different temperature or strain ranges. Therefore, as stated in section 3.6, a promising approach for SMA control is to implement gain scheduling into the PID controller. By making the controller gain proportional to the strain–temperature derivative of the model, the PID controller could achieve more consistent responses while maintaining a relatively simple structure.

The downside of this approach is that it still relies on a computationally intensive and complex Preisach model. Although the controller structure itself (PID with gain scheduling) remains simple, the underlying hysteresis model used to compute the varying gain does not. In addition to a strain estimate, the model also requires a temperature estimate. Note that including a load or stress estimate would improve accuracy, but is not strictly necessary, since the strain–temperature derivative varies only slightly across different load ranges.

For implementation in a robotic tentacle with multiple actuators, one option is to equip each actuator with its own thermocouple. Alternatively, a single thermocouple could be placed on one actuator (e.g., the bottom actuator), and a temperature model could be used to estimate the temperatures of the remaining actuators based on the input current and the measured temperature. However, both approaches increase system complexity. Therefore, since one of the goals of this research is to demonstrate the feasibility of SMAs in soft robotics, the robotic tentacle described in the next chapter uses the PID controller without the gain scheduling approach described in section 3.4.

3.8. Chapter Summary

To move toward the goal of designing and controlling a biomimetic tentacle, this chapter first focused on controlling the strain of a single actuator. Two controllers were proposed for this task: a PID controller and a FF–FB controller. Since the FF–FB controller requires a model for its feedforward component, the Preisach hysteresis model (described in section 2.3) was identified. To obtain this model, three different calibration methods were investigated.

The first method was a temperature-control-based calibration approach (see Appendix D), which resulted in a poor model due to drift in the thermocouple readings. A second PWM-based calibration method (see Appendix E) addressed this issue by using a steady-state temperature model and estimating the temperature during the experiment in a post-processing step. The resulting model showed good agreement with experimental data for cooling trajectories, but poor agreement for heating trajectories. This was caused by the FORC increments being measured during cooling only. Therefore, a third calibration approach was developed, in which the Preisach density function was identified for both heating and cooling, resulting in a dual Preisach model. This model showed improved results and very good agreement for major hysteresis loops, but in some cases still poor agreement for minor loops.

Both controllers were evaluated based on their tracking performance using step and sinusoidal reference signals. The PID controller showed surprisingly good tracking performance, with the main drawback being an inconsistent temperature–strain response across different strain ranges. The FF–FB controller showed worse tracking performance, mainly due to model inaccuracies, a slower response caused by the cascaded control structure, and sensitivity to the rate of strain change. Its main advantage, however, was a more consistent response across different strain ranges. A possible improvement would therefore be a PID controller with gain scheduling, where the gains vary with the strain–temperature derivative predicted by the model.

Based on a set of specified criteria, it was ultimately concluded that a PID controller is the most suitable choice for a soft robotic tentacle. This is mainly due to its simplicity, limited number of required state estimates, lower implementation complexity, and strong tracking performance. To further pursue the research objective of designing and evaluating a complete soft robotic tentacle, the next chapter presents the full tentacle design along with the performance of the PID controller.

4

Control of Multiple Antagonistic SMA Actuators

This chapter aims to address the research goals proposed in section 1.3 by designing a soft robotic tentacle, implementing the PID controller proposed in the previous chapter, and evaluating the performance of the system. Therefore, this chapter tries to answer how a desktop-scale tentacle can be designed and what level of tracking performance can be achieved using a PID controller. The mechanical design, electronics, and control architecture are discussed in section 4.1. The control strategy, including state estimation and the design of the PID controller, is presented in section 4.2. The controller results and an evaluation of the design and controller performance is provided in section 4.3. Based on this evaluation, potential improvements are discussed in section 4.4. Finally, a summary of this chapter is provided in section 4.5.

4.1. Tentacle Design and Control Architecture

As briefly stated in section 1.3, the research questions can be addressed by designing a soft robotic tentacle composed of multiple rotating sections. The main goal of the tentacle is to evaluate the feasibility of SMAs within a biomimetic structure and to assess the performance of the PID controller. Therefore, a relatively simple design with three 3D-printed sections is sufficient for this purpose. 3D printing was chosen as it enables easy fabrication and allows for straightforward modifications to the design.

4.1.1. Mechanical Design and Electronics

A CAD representation of the designed tentacle is presented in Figure 4.1a. It consists of a base, where the electronics are mounted, and three sections $i = 1, \ldots, 3$, each with two axes x_i and y_i that can rotate by means of a Hooke's joint [25], also known as a universal joint. A detailed view of this mechanism is provided in Figure 4.1b, showing how each section can rotate about two axes. Each section is limited to a maximum rotation of 30°, resulting in a maximum overall deflection of 90° at the tip in any direction. The reason for choosing four actuators—rather than three, as in the example from section 1.3—is that this configuration makes each axis independent. For example, the antagonistic pair of SMAs connected to x_1 can be actuated without affecting the load on the pair connected to y_1. This simplifies controller tuning and evaluation. Furthermore, a different Nitinol actuator (not shown in Figure 4.1a) with a larger spring diameter (approximately 8 mm) and a thicker wire diameter (1.25 mm) is used in the tentacle due to its higher force output. Various images of the fabricated tentacle are provided in Appendix G.

To measure the angle of each of the six axes, the center piece of the joint (pink) contains a magnet that moves closer to or further away from a Hall sensor embedded in the section. This approach was chosen for its simplicity and the low mass of the magnet–sensor combination. Other options, such as encoders or potentiometers, were considered; however, encoders are relatively expensive, and potentiometers typically introduce friction that could affect the system's behavior. By using a low-cost Hall effect sensor and reading its analog output with a microcontroller, the angles of the sections can be measured in a

straightforward manner. A drawback of this approach is the limited accuracy and potential noise in the analog signal. However, since the primary goal is to evaluate the controller performance and the feasibility of SMAs, high measurement accuracy is not required.

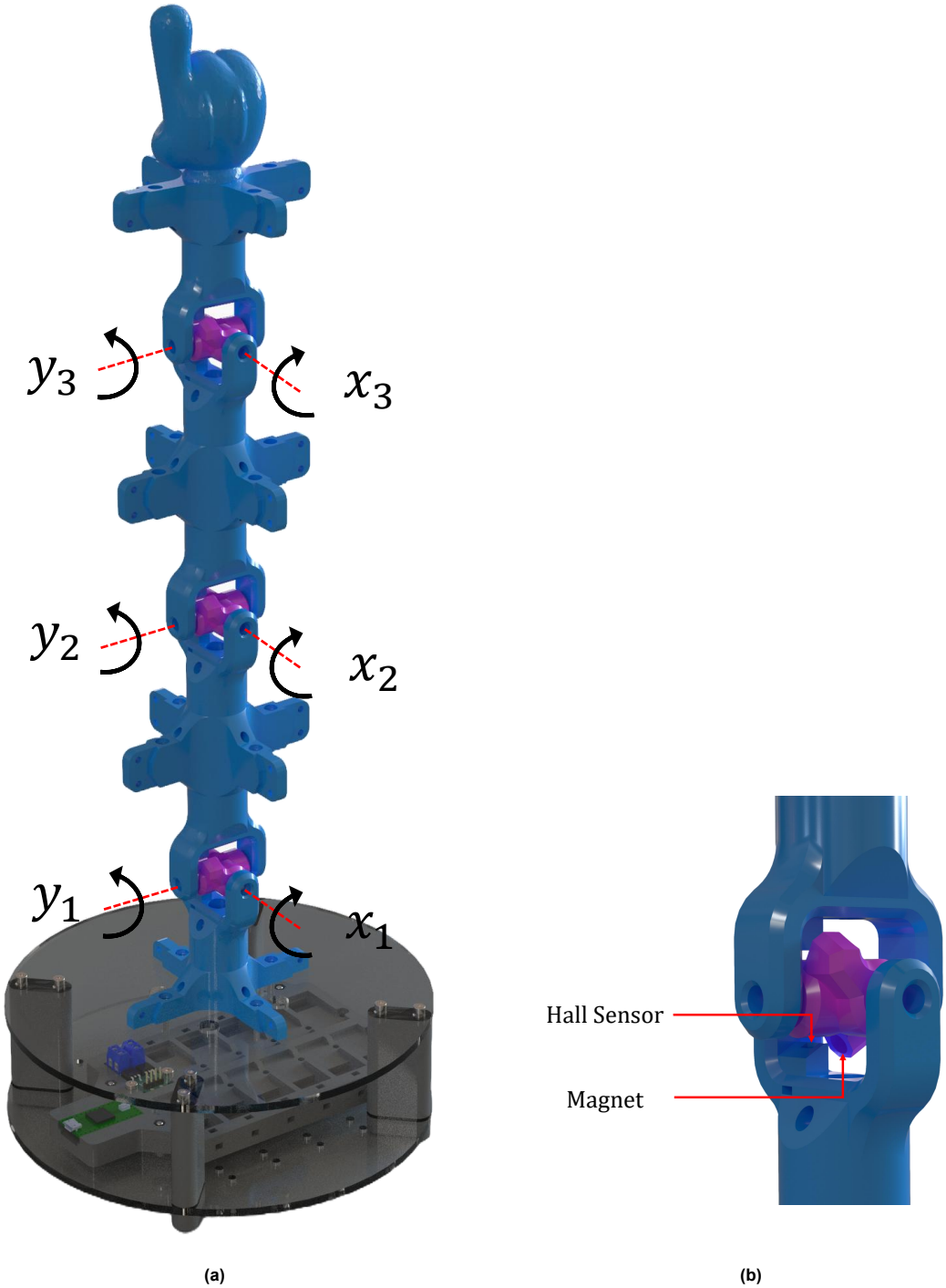


Figure 4.1: CAD representations of the designed tentacle. (a) View of the entire tentacle, including its base and three independent sections. The axes x_i and y_i for $i = 1, \ldots, 3$ denote the angles of sections 1–3. (b) Detailed view of a center piece (pink). A magnet is inserted into the center piece, which moves closer to or farther from the Hall sensor embedded in the section (blue) as the section rotates.

The tentacle contains electronics similar to those used in the test bench described in section 2.4: a Teensy 4.0 microcontroller [30] and 12 PWM modules containing dual AOD4184A MOSFETs [3]. To

simplify the design, and since they are not required for the control objectives, the thermocouple, voltage divider, and current sensor are omitted in this setup. The Hall effect sensor used is the OH49E Hall Effect Sensor [26], which was primarily selected due to its availability at the time of implementation.

4.1.2. Control Architecture

The control architecture of the tentacle, shown in Figure 4.2, is similar to that of the test bench. The microcontroller reads the Hall sensor data, applies filtering, and transmits the measurements to Python via serial communication at 100 Hz. In Python, multiple PID controllers (as described in section 3.1) run at 40 Hz and send their outputs as duty cycle commands back to the microcontroller, which then generates the corresponding PWM signals. These signals are sent to the MOSFETs on the PWM modules, resulting in actuation of the SMAs. The PWM modules are powered by a single-cell lithium-ion battery, which is omitted from Figure 4.2 for clarity. This architecture enables full control of the system through Python once the microcontroller has been programmed.

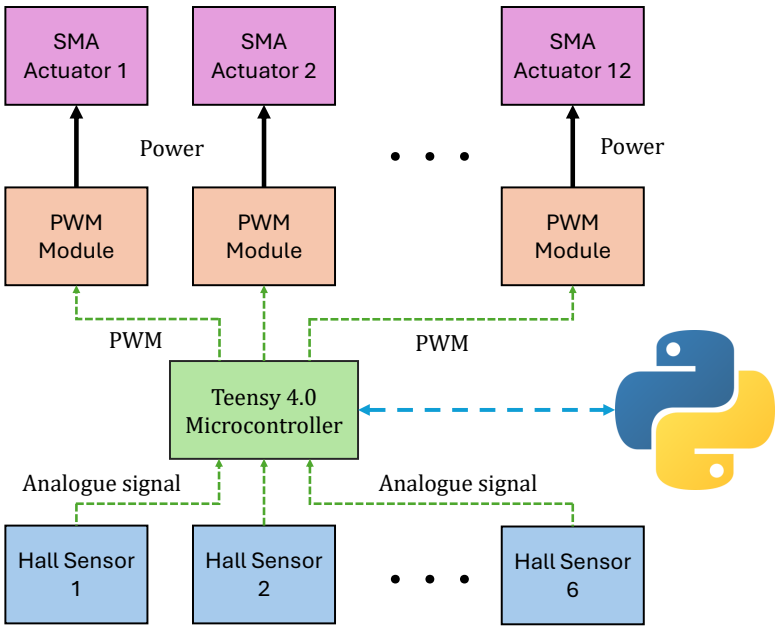


Figure 4.2: Schematic of the control architecture of the designed tentacle. The microcontroller (green) reads the analog Hall sensors (blue), filters the data, and transmits it to Python over a serial connection. Python sends duty cycle commands to the microcontroller, which generates PWM signals that are passed to the PWM modules (orange), ultimately driving the actuators (pink). The PWM modules are powered by a lithium-ion battery (not shown for clarity).

4.2. Control Strategy

As stated in section 4.1, each section contains two pairs of antagonistic actuators that are independent from one another, meaning that actuating one pair does not directly affect the load, and therefore the behavior, of the other pair. As a result, each of the six antagonistic actuator pairs within the tentacle can be treated as an independent system. This allows for the design of a single controller structure that can be applied to each of the six axes. It should be noted, however, that different system responses can be expected between the axes, as the load on each axis—caused by the mass of the tentacle during deflection—varies between the lower and upper sections. Consequently, different controller gains may be required across the sections to achieve similar performance.

4.2.1. PID Controller Design

The main difference between controlling a tentacle section and controlling the SMA in the test bench is that the test bench uses a single actuator under a fixed load, whereas each axis of a section consists of a pair of antagonistic actuators that depend on each other for motion. Since the PID controller proposed

in the previous chapter has a single output, but two actuators must be driven, this creates a problem. Therefore, a new strategy is required to control the angle of each axis.

One possible approach is to assign a separate PID controller to each actuator, resulting in a total of 12 PID controllers for the entire tentacle. However, this introduces an issue: for a given reference angle of a section, multiple combinations of PID outputs can result in that reference. For example, if the reference for x_1 is $0°$, this could be achieved by setting both actuator duty cycles to 50%, but also by setting both to 100%. In general, infinitely many actuator combinations exist for an antagonistic pair, resulting in non-unique controller outputs, actuator conflict, and instability.

Therefore, PID control strategies in the literature often treat an antagonistic pair of actuators as a single system, where the actuator outputs are coupled. For example, the method described by [15] controls an antagonistic pair of SMAs in a biomimetic finger by activating only one actuator depending on the PID output. Additionally, to increase stiffness and stabilize the finger position, both actuators are activated within a small error band using fuzzy logic that maps the single PID output to two PWM signals.

To ensure a unique control action and improve stability, each section—containing two actuators, SMA A and SMA B—is therefore controlled by a single PID controller, as shown in Figure 4.3. The error signal is computed as the difference between the reference angle θ_r and the measured angle θ_m of the section. The control input $u_A(t)$ is defined as the sum of the PID output $u(t)$ and a constant offset K_{ff}, while $u_B(t)$ is defined as the difference between $u(t)$ and K_{ff}. The signals $u(t)$, $u_A(t)$, and $u_B(t)$ represent PWM duty cycles and are constrained within the interval 0–100%.

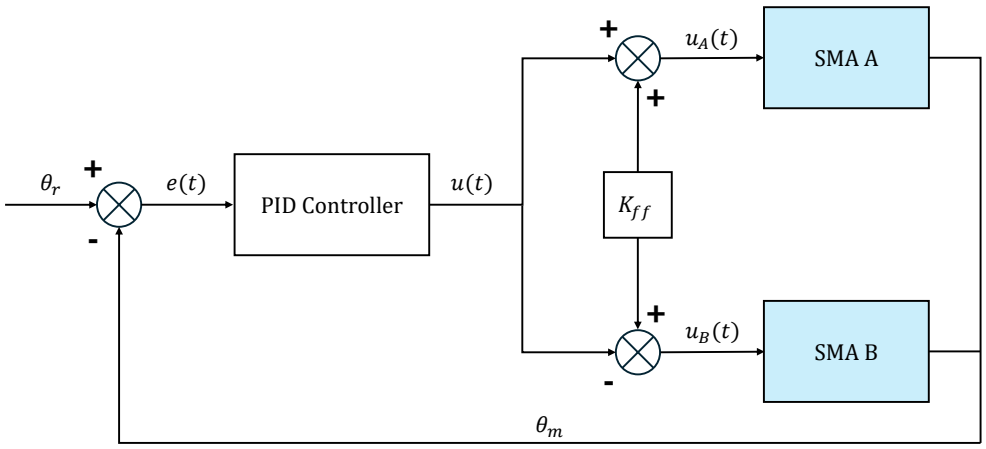


Figure 4.3: Control scheme of the proposed PID controller. The reference angle θ_r is compared with the measured angle θ_m to compute the error $e(t)$, which is used by the PID controller to generate $u(t)$. This signal is combined with the feedforward term K_{ff} to produce the actuator inputs $u_A(t)$ and $u_B(t)$ for the antagonistic SMA pair.

Using this control strategy, the output of the PID controller is mapped to two actuator inputs. In addition, similar to the method described in [15], the stiffness—and therefore the stability—of the system is increased by actuating both actuators when $e(t)$, and thus the PID output, becomes small. In the proposed approach, this is achieved by adding a constant feedforward term K_{ff} to the control signals $u_A(t)$ and $u_B(t)$. Without K_{ff}, no heating occurs when the error approaches zero, resulting in low stiffness of a section and consequently less stable system behavior. The value of K_{ff} can be tuned empirically, but it must remain below the duty cycle corresponding to a steady-state saturated austenite temperature to prevent overheating of the actuators. The resulting control inputs are defined as

$$u_{\mathrm{PID}}(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d\, \frac{de(t)}{dt} \tag{4.1}$$

$$u_{A,B}(t) = K_{ff} \pm u_{\text{PID}}(t), \tag{4.2}$$

where the '+' sign corresponds to actuator A and the '-' sign to actuator B.

4.2.2. State Estimation

To generate the error signal, the angle of each section is estimated using Hall effect sensors. As stated in section 4.1, each section contains a sensor that produces an analog signal depending on the distance to a magnet, and therefore on the angle of that section. Since high accuracy in the angle estimation is not required to achieve the research objectives, a simple estimation method based on a third-degree polynomial fit is used. Four data points were obtained visually and fitted as shown in Figure 4.4.

Note that fitting a third-degree polynomial to four data points results in a perfect interpolation and is therefore not a physically accurate representation of the true angle–sensor relationship. However, it provides a fast and simple way to estimate the angles, with the main purpose being to linearize the non-linear response of the Hall sensors rather than to achieve high precision. The same polynomial mapping is used for each section to convert the analog sensor readings into angles.

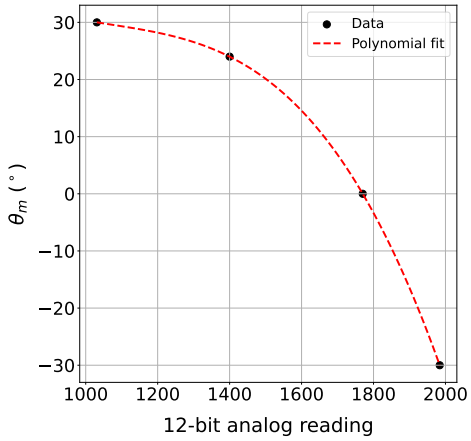


Figure 4.4: Relationship between the angle of a tentacle section and the Hall effect sensor reading measured by the microcontroller. The four data points (black) are obtained visually, and the fitted curve (red) is a third-degree polynomial.

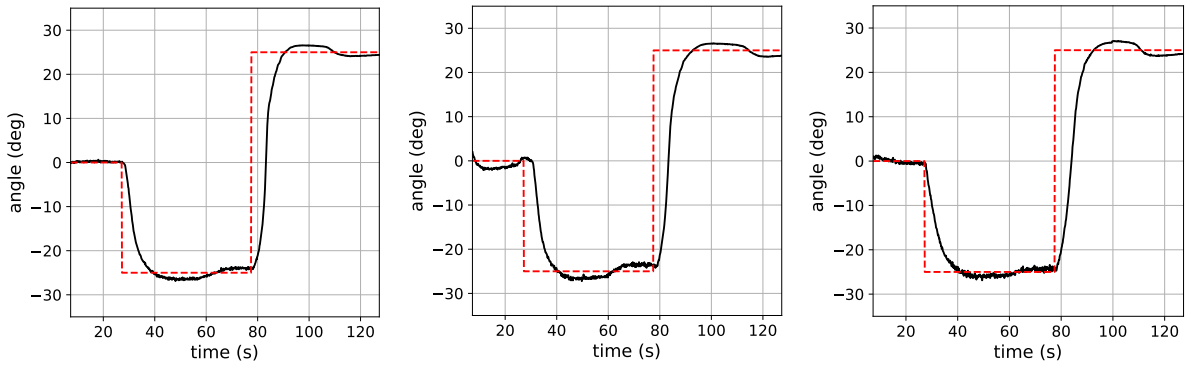
Moreover, the noisy Hall effect sensor readings are filtered using the same running average filter (Equation 2.27) that was previously applied to the current sensor and thermocouple measurements in the test bench. A sample window of 500 measurements is used, as this was empirically found to provide a good trade-off between noise reduction and response speed.

4.3. Controller Results and Performance Evaluation

Similar to the method described in chapter 3, the performance of the system is evaluated based on its tracking performance using large step input references, incremental step references, and sinusoidal references. To answer the proposed research questions and achieve the research objectives, the primary goal of this evaluation is to assess both the tracking performance of the controller and the overall performance of the tentacle. In this way, the feasibility of using SMAs as actuators within a biomimetic tentacle can be evaluated.

4.3.1. Controller Evaluation

In the first experiment, the system behavior is evaluated for large step inputs by driving all x-axes to $-25°$ and subsequently to $25°$, while keeping all y-axes at a reference of $0°$. This results in a side-to-side swaying motion of the tentacle. The responses of the three x-axes are shown in Figure 4.5.



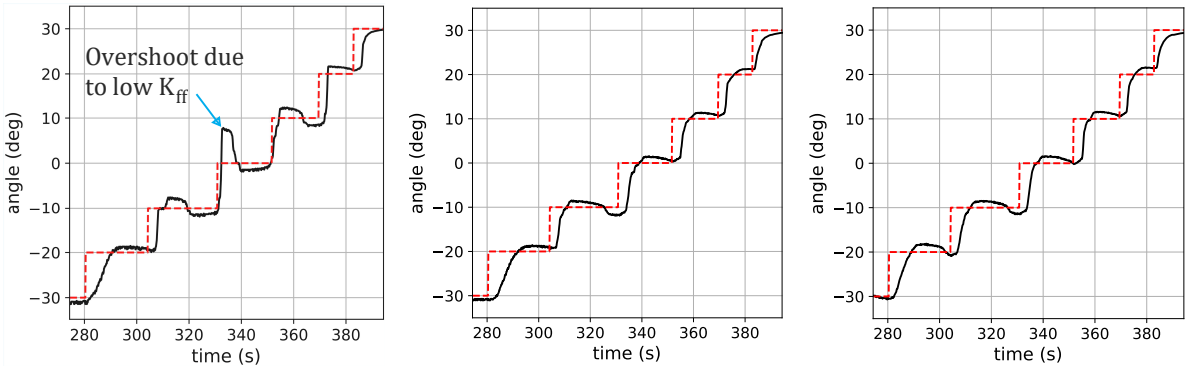
(a) Axis x_1. **(b)** Axis x_2. **(c)** Axis x_3.

Figure 4.5: Responses of the x-axes for reference inputs of $-25°$ and $25°$ during the step experiment. The reference signal is shown in red, while the black curves represent filtered Hall sensor measurements. Subfigures **(a)**, **(b)**, and **(c)** correspond to axes x_1, x_2, and x_3, respectively.

For simplicity, all PID gains and the controller parameter K_{ff} are chosen to be identical for each axis. These values are empirically tuned based on the response of the lower section of the tentacle. The gains are selected to achieve a trade-off between fast response and overshoot, which explains the visible overshoot for all x-axes in Figure 4.5. Reducing the proportional and integral gains decreases this overshoot, but at the cost of poorer tracking of dynamic references, such as the sinusoidal inputs discussed later. Furthermore, it is observed that increasing K_{ff} reduces oscillations around the setpoint, likely due to increased stiffness at higher actuator temperatures, which is consistent with the method described in [15].

In the previously described test bench, the system response (see section 3.4) could be effectively damped by increasing the derivative gain of the PID controller. By briefly applying a high current—and thus a large heat input—a fast response could be achieved while maintaining low overshoot. However, this approach is not directly applicable to the tentacle due to limitations of the MOSFETs on the PWM modules, which overheated during experiments at high values of K_{ff} or large derivative gains. This overheating is caused by the larger diameter SMAs used in the tentacle, which require higher currents to reach the same temperatures as the smaller SMA used in the test bench. To prevent damage to the MOSFETs, the derivative gain was kept low. As a result, the overshoot observed in Figure 4.5 could likely be reduced by increasing the derivative term, but this was not tested due to hardware constraints.

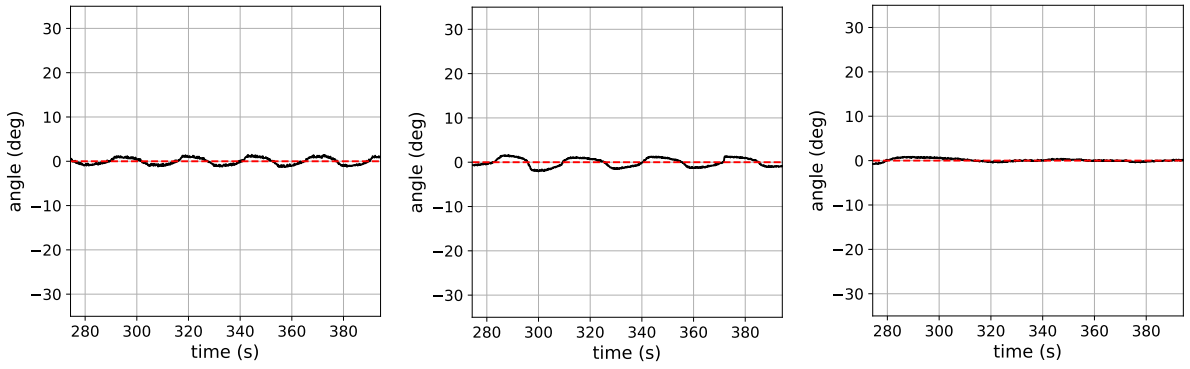
The effect of the relatively low value of K_{ff} becomes particularly clear in the incremental step response experiment shown in Figure 4.6 and Figure 4.7. Similar to the large step input experiment, the y-axis references are kept at $0°$, while the x-axis references are increased incrementally from $-30°$ to $30°$ in steps of $10°$. As shown in Figure 4.6a, a significant overshoot occurs when stepping toward $0°$. This can be explained by the preceding motion: during earlier steps, the tentacle is deflected to the left, causing the right-side actuators of x_1, x_2, and x_3 to operate at higher temperatures due to increased load, while the left-side actuators remain relatively cool. When the reference reaches $0°$, the torque direction of axis x_1 reverses, and the load suddenly shifts to the colder, less stiff left actuator. Since this actuator requires time to heat up and generate stiffness, an overshoot occurs. This effect is most pronounced for x_1, as it experiences the largest torque change due to the mass and inertia of the tentacle. Although this overshoot could be reduced by increasing K_{ff}, this was not performed due to the previously mentioned overheating issues of the MOSFETs.



(a) Axis x_1. **(b)** Axis x_2. **(c)** Axis x_3.

Figure 4.6: Responses of the x-axes for reference inputs from $-30°$ to $30°$ in $10°$ increments during the incremental step experiment. The reference signal is shown in red, while the black curves represent filtered Hall sensor measurements. Subfigures **(a)**, **(b)**, and **(c)** correspond to axes x_1, x_2, and x_3, respectively.

Overall, the results show that the PID controller is able to reliably track step inputs, albeit with some oscillations around the reference once it is reached. To illustrate these oscillations, the responses of the three y-axes during the incremental step experiment are shown in Figure 4.7, where each axis tracks a constant reference of $0°$. It can be observed that the oscillations around the setpoint are largest for the lower section and smallest for the upper section. A possible explanation is the disturbance caused by the momentum of the tentacle as the x-axes move between setpoints. This disturbance is more pronounced in the lower section due to the larger torque present in that section. Increasing K_{ff} could potentially reduce these oscillations by increasing actuator stiffness, although this has not been verified.



(a) Axis y_1. **(b)** Axis y_2. **(c)** Axis y_3.

Figure 4.7: Responses of the y-axes for reference inputs of 0 during the incremental step experiment. The reference signal is shown in red, while the black curves represent filtered Hall sensor measurements. Subfigures **(a)**, **(b)**, and **(c)** correspond to axes y_1, y_2, and y_3, respectively.

Another important observation is that, for each axis, $0°$ represents an unstable equilibrium, meaning that even small disturbances cause the system to move further away from this position. This behavior is caused by gravity as well as the load introduced by the actuators of the orthogonal axis. For example, when actuating axis y_1 while keeping x_1 unactuated (cold), axis x_1 will rotate toward either $-30°$ or $30°$, depending on its initial condition. This indicates that the axes are not fully independent, as suggested in section 4.2, and helps explain the oscillations observed in Figure 4.7. Increasing K_{ff} raises the stiffness of the actuators and thus the vertical load on each axis, which may further amplify this tendency to move away from the unstable equilibrium.

In the final reference tracking experiment, the lower and middle sections of the tentacle are actuated to

produce the rotating motion shown in Figure 4.8, while the top section remains unactuated. This results in a motion where the tip of the tentacle points upward and follows a circular trajectory. Consequently, the Hooke's joints in the lower and middle sections follow a conical motion. Although the resulting trajectories are not purely sinusoidal but are instead composed of trigonometric functions, they are referred to as sinusoidal for simplicity.

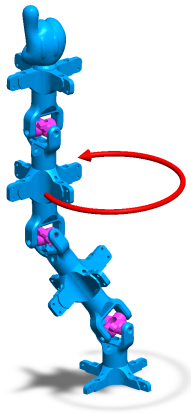


Figure 4.8: Reference trajectory used in the sinusoidal tracking experiment. The axes of the lower and middle sections follow the inverse kinematic equations provided in Equation 4.5, with opposite phase to produce the shown motion.

To obtain the reference trajectory shown in Figure 4.8, we first define the motion of the lower section, corresponding to axes x_1 and y_1. The kinematics of a Hooke's joint are described in [8] and can be expressed using rotation matrices as

$$\mathbf{u} = R_x(x_1)\, R_y(y_1)\, \mathbf{e}_z, \tag{4.3}$$

where x_1 and y_1 are the joint angles, and $\mathbf{e}_z = [0\ 0\ 1]^T$ denotes the reference direction. Based on this formulation, a conical motion can be represented by the unit vector

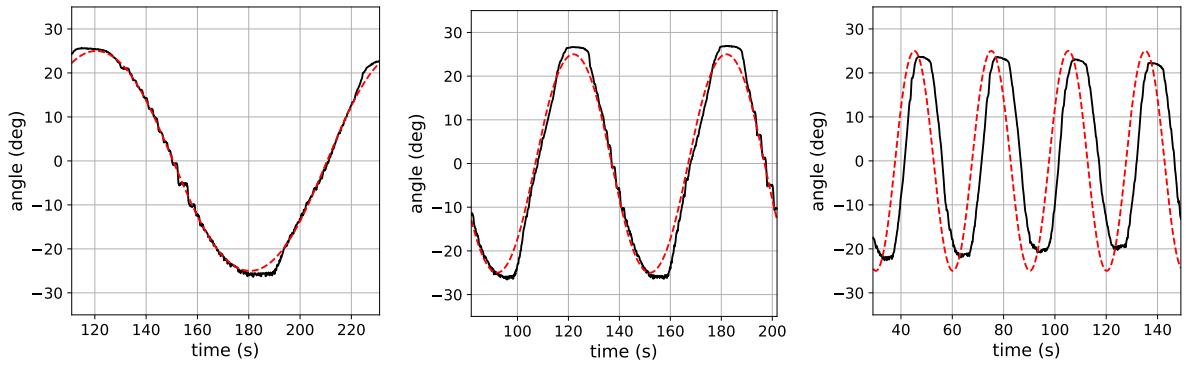
$$\mathbf{u}(t) = \begin{bmatrix} \sin\beta\cos(\omega t) \\ \sin\beta\sin(\omega t) \\ \cos\beta \end{bmatrix}, \tag{4.4}$$

where β is the cone angle and ω is the angular frequency. By solving the inverse kinematics, the corresponding trajectory for the lower section is obtained as

$$\begin{bmatrix} x_1(t) \\ y_1(t) \end{bmatrix} = \begin{bmatrix} \arctan\big(-\tan\beta\sin(\omega t)\big) \\ \arcsin\big(\sin\beta\cos(\omega t)\big) \end{bmatrix} \tag{4.5}$$

The kinematics for the middle section—that is, axes x_2 and y_2—follow the same formulation but are driven in opposite phase to create the desired coordinated motion. Therefore, the trajectories are defined as $x_2 = -x_1$ and $y_2 = -y_1$.

The responses of x_1 during the sinusoidal reference tracking experiment are shown in Figure 4.9 for increasing frequencies of $\frac{1}{120}$ Hz, $\frac{1}{60}$ Hz, and $\frac{1}{30}$ Hz. The responses of y_1, x_2, and y_2 are provided in Appendix H and exhibit similar behavior due to the symmetry of the tentacle. In general, the controller tracks slowly varying references well, but performance degrades at higher frequencies due to the limited bandwidth of the SMA actuators.



(a) x_1 response at $\frac{1}{120}$ Hz. **(b)** x_1 response at $\frac{1}{60}$ Hz. **(c)** x_1 response at $\frac{1}{30}$ Hz.

Figure 4.9: Response of axis x_1 to sinusoidal reference inputs at three excitation frequencies: **(a)** $\frac{1}{120}$ Hz, **(b)** $\frac{1}{60}$ Hz, and **(c)** $\frac{1}{30}$ Hz. The reference signal is shown in red, while the black curve denotes the filtered Hall sensor measurement.

As stated before, the proportional and integral gains were tuned relatively high to improve tracking performance for the sinusoidal reference. Reducing these gains decreases the overshoot in the step responses, but degrades tracking performance for sinusoidal references, particularly for rapidly changing setpoints. This effect is clearly visible in Figure 4.9b, where the deviation between the measured angle and the reference is largest near the peaks and troughs of the sinusoidal signal. A relatively high integral gain allows the controller to quickly compensate for small errors, improving tracking of dynamic references. However, it also introduces undesired integral windup during step inputs.

Overall, the PID controller is easy to tune, shows consistent response, and achieves acceptable tracking performance. Its performance is, however, limited by the relatively low value of K_{ff}, which likely contributes to the oscillations observed around constant setpoints. In addition, the controller is unable to fully suppress the overshoot in the step responses, although this may be due to limitations in the power electronics rather than the controller itself. Finally, because the controller does not explicitly account for system dynamics, large overshoots can occur when the load on an actuator changes abruptly due to variations in the tentacle configuration.

4.3.2. Design Evaluation

As discussed, the performance of the system is likely limited by the relatively low value of K_{ff}, which results in reduced stiffness and consequently some instability and oscillations around the reference. In addition, the derivative gain is kept low, whereas a higher value could have reduced the overshoot observed in the step response experiments. Therefore, the main improvement to the design would be to use different MOSFETs or find a method to prevent them from overheating. Small aluminum heat sinks were added to the MOSFETs to increase natural convection. However, this did not resolve the issue.

According to the datasheet, a single MOSFET reaches a temperature of approximately $70°$C when a pulse drain current of 10 A is applied, where the pulse drain current corresponds to the continuous current during PWM actuation. Since the SMAs used in the tentacle draw around 2 A at steady-state and up to approximately 6 A during high PID effort (which in practice occurs only briefly), the MOSFETs should, in theory, not overheat—especially considering that each PWM module uses two MOSFETs in parallel. As the datasheet does not specify the frequency of the pulse drain current, the initially used PWM frequency of 1 kHz was reduced to 20 Hz to investigate whether switching losses were responsible for the overheating. However, this change did not noticeably affect the temperature of the MOSFETs.

Furthermore, large steady-state temperature differences were observed between actuators actuated with the same duty cycle, suggesting a more fundamental issue in the electronics. Upon closer inspection of the datasheet, it was concluded that the likely cause of both the overheating and inconsistent actuator response is the low gate voltage applied to the MOSFETs. The microcontroller used in this project

outputs PWM signals at 3.3 V. Although the MOSFETs can be driven at this voltage, their on-resistance increases extremely quickly for low gate voltages, as shown in Figure 4.10. A high on-resistance leads to a large voltage drop across the MOSFET and increased power dissipation, ultimately causing overheating. Due to time constraints, replacing the MOSFETs or increasing the gate drive voltage was not performed.

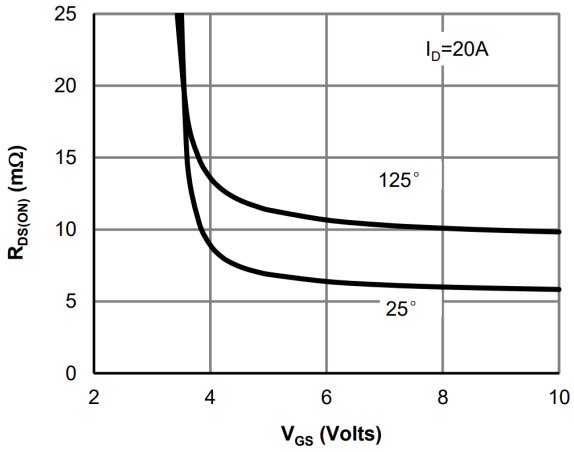


Figure 4.10: On-resistance as a function of gate voltage for the MOSFETs used in the tentacle. The figure shows a significantly higher resistance at the low gate voltage provided by the microcontroller, explaining the observed overheating. Reproduced from [3].

4.4. Future Improvements

Although the results of the tentacle demonstrate that SMAs can be used in a controlled setting, the response remains relatively slow and shows oscillations and overshoot. A straightforward improvement would be to upgrade the power electronics, allowing for higher values of K_{ff} and thus increased system stiffness. However, this would also lead to higher power consumption, as the actuators draw significant current when reaching the setpoint. To solve this, the tentacle design could be modified to use three actuators per section instead of four.

In addition, the slow response of the system could be improved by introducing active heat transfer, rather than relying solely on natural convection. One approach is to enclose each actuator in a channel or tube and force air through it, as illustrated in Figure 1.5. Another approach—discussed earlier in section 1.1—is to use an SMA embedded in a silicone tube filled with non-conductive oil, enabling conductive heat transfer. A drawback of this method is that the oil heats up over time, reducing heat transfer efficiency, lowering the bandwidth, and altering the system behavior.

Therefore, instead of enclosing the SMA in a fixed volume of oil, the entire tentacle could potentially achieve improved bandwidth in underwater applications, where the surrounding water provides conductive heat transfer. A drawback of this approach is the electrical conductivity of water—especially seawater with high salt content—as well as sensitivity to variations in ambient water temperature. In addition, the tentacle is equipped with relatively thick SMAs, which have a lower surface-area-to-volume ratio compared to thinner wires, limiting conductive heat transfer. An alternative approach to improve bandwidth is therefore to use multiple parallel springs with smaller diameters and/or wire thicknesses. While such methods are reported in the literature, they increase the mechanical design complexity.

An advantage of using SMAs in robotics, compared to conventional actuators such as electric motors, is the relatively low number of required components, resulting in a simpler overall system. Motion can be achieved by integrating springs directly into the structure, whereas electric motors often require additional components such as gearboxes to provide sufficient torque. In the designed tentacle, however, the mechanical complexity is increased by the inclusion of Hall effect sensors and magnets for angle estimation. An alternative method for measuring the strain of an actuator—and thus the angle of a section—is to measure its electrical resistance. The strain can be estimated based on the resistance,

although this relationship is hysteretic, as discussed in subsection 2.1.2. Such methods are reported in the literature, for example in [21]. However, characterizing the strain–resistance relationship requires large effort.

Another possible improvement is to implement a method for estimating the temperature of each actuator in the tentacle. This would allow the (PID) controller to detect when an SMA has reached its austenite saturation temperature and stop undesired overheating. As a result, the controller could be tuned more aggressively without risking overheating and damaging the actuators. In the current implementation, the PID output is constrained within a fixed duty cycle range to prevent overheating, where the upper limit is based on the duty cycle corresponding to a steady-state austenite saturation temperature. The drawback of this approach is that the controller cannot briefly apply high heating inputs, limiting its ability to respond quickly to disturbances. However, estimating the actuator temperature while maintaining a simple mechanical design is challenging, as adding a thermocouple to each actuator would significantly increase system complexity. An alternative approach is to measure the temperature of a single actuator and estimate the remaining actuator temperatures using a model. The downside of this approach is that it requires a complex temperature model.

From a control design perspective, the accuracy and tracking performance achieved with the PID controller are promising. However, a key limitation is that the controller has no knowledge of the system dynamics and relies on only four tuning parameters. While this simplicity results in consistent and robust behavior, it restricts further performance improvements, particularly in reducing oscillations around the setpoint and overshoot. Therefore, a more advanced control strategy that accounts for the system dynamics is required to achieve improved tracking performance.

However, as concluded in chapter 3, implementing a model-based control strategy is challenging due to the complex hysteretic and non-linear behavior of SMAs. In addition, such approaches typically require a larger number of state estimates—such as temperature and actuator load—which increases the complexity of the mechanical design. As illustrated in Figure 3.15, the hysteretic behavior of the tested SMA is also rate-dependent, showing significantly different temperature–strain responses at higher heating rates. Moreover, more advanced control strategies often involve cascaded structures, for example combining a model block with a temperature control loop, which further increases system complexity.

As a result, the combination of many required state estimates, the complexity of the underlying model, and the more complicated control architecture makes the implementation of model-based control challenging. Alternative approaches reported in the literature include dynamic models that predict SMA force output, as well as artificial intelligence (AI) methods based on machine learning to identify patterns between inputs and actuator response. These approaches were beyond the scope of this research, but may provide a promising direction for improving the performance of the designed tentacle in future work.

4.5. Chapter Summary

This chapter evaluated the feasibility of using SMAs as actuators in a robotic tentacle with respect to controllability and performance. This was done by designing and constructing a tentacle composed of three sections, where the two orthogonal angles of each section are measured using a Hall effect sensor and a magnet. The PID controller proposed in chapter 3 was implemented for each of the six axes. Since the designed system contains two actuators per axis rather than one, the output of the PID controller was mapped to two actuator inputs.

Three experiments were performed to evaluate the controller performance: a large step experiment, an incremental step experiment, and a sinusoidal reference tracking experiment. Since the controller was tuned with relatively high proportional and integral gains, the tracking performance—especially for slowly varying sinusoidal references—was good. For step inputs, however, the response showed overshoot, which is likely caused by the low value of K_{ff} and the low derivative gain. These were intentionally kept low to prevent overheating of the MOSFETs in the system. Furthermore, oscillations around constant setpoints were observed, likely due to limited actuator stiffness, which is also related to the low value of K_{ff}. Therefore, the performance of the PID controller is likely limited by the power electronics.

All in all, the PID controller proved to be a reliable and robust way of controlling the tentacle. The SMAs were able to bring the tentacle to its desired pose, and the tracking performance was adequate. However, achieving further improvements in controller accuracy remains challenging due to the complex behavior of SMAs and the more advanced control strategies required to handle this. While the simplicity of the PID controller makes it easy to implement, it also limits future improvements, as the controller does not include any knowledge of the system dynamics.

5

Conclusion

This research evaluated the control and feasibility of SMAs in a biomimetic tentacle. This was achieved through a literature review, in which existing control strategies were explored. From this review, it was found that commonly used approaches rely on a feedforward–feedback (FF–FB) strategy, which requires accurate modeling of the SMA behavior. To simplify the control problem, two controllers—a PID controller and an FF–FB controller—were first evaluated on a single actuator. After observing that the PID controller provided better performance, it was extended and implemented in a full prototype tentacle, whose performance was assessed through experimental testing.

To conclude the findings of this research, the conclusions are structured according to the three research goals and their associated research questions, as introduced in section 1.3.

Design and Construction of a Prototype

The design and construction of the prototype tentacle proved to be a relatively straightforward task. Since actuation using SMAs can be achieved by arranging them antagonistically, the mechanical design is simpler compared to traditional robotic arms driven by electric motors. For the prototype, 3D printing provided a flexible manufacturing method, and the power electronics consist of a set of MOSFETs driven by PWM signals. Therefore, it is concluded that this design goal has been successfully achieved.

However, improvements can be made in the state estimation approach. In the current design, this is achieved by measuring the distance between Hall effect sensors and magnets integrated within each section, followed by filtering of the measurements. Although this method provides reliable angle estimates, it requires calibration of the non-linear relationship between the sensor output and the angle, which is time-consuming. In addition, it increases the mechanical complexity of the system. In future work, the strain of the actuators—and thus the section angles—could be estimated by measuring the electrical resistance of the SMAs. This can be done using the existing wiring used for actuation, thereby simplifying the mechanical design. The drawback of this approach is that it requires identification of a hysteretic strain–resistance relationship, which can be complex.

For more advanced control strategies, as the one discussed in subsection 3.5.3, additional state estimates—such as actuator temperature and/or strain—are typically needed. This requires extra sensors or estimation methods, increasing the overall system complexity. As a result, implementing such advanced controllers in a biomimetic tentacle—despite their potential for improved performance—remains a challenging task.

Development of a Control Strategy

The literature study revealed that most advanced control strategies use a FF-FB strategy, where the feedback component is a PID controller and the feedforward component a temperature prediction from a model. Therefore, two control strategies were implemented in chapter 3 to control a single SMA actuator: a pure PID controller and a FF-FB controller. The model used in the FF-FB controller was a Preisach model which was inverted to output a temperature reference.

Based on the system analysis of both controllers, it was concluded that a PID controller was the most suitable choice for controlling the prototype due to its superior tracking performance, simplicity and low required number of state estimates. To implement the PID controller in the prototype, it was expanded to map its output to two actuator inputs rather than one by implementing an additional tuning parameter.

Overall, the PID controller performed consistently, was able to track references reasonably well, and was straightforward to tune. However, it struggled with load fluctuations, as it has no knowledge of the system dynamics, leading to large overshoots when the actuator load changes abruptly due to variations in the tentacle configuration. In addition, oscillations were observed around constant setpoints. These observations should be interpreted with caution, as they may result from limitations of the power electronics in the designed prototype rather than the controller itself. Furthermore, the lack of system knowledge and the limited number of tuning parameters make it difficult to further improve performance. Achieving better tracking performance will likely require a more advanced control strategy that accounts for the system dynamics.

Implementing such an advanced controller, however, remains a challenging task. As shown in chapter 3, the calibration procedure of the model must be performed carefully, is complex, and is time-consuming. In addition, the hysteretic temperature–strain relationship of SMAs was found to be rate-dependent, which further complicates the modeling required for control. Moreover, advanced controllers typically require a larger number of state estimates. Therefore, a PID controller is a suitable choice when reliable tracking performance is sufficient and high accuracy is not critical. For applications requiring higher accuracy, further research is needed, as the behavior of SMAs remains complex and difficult to control.

In terms of settling time and bandwidth, the prototype showed relatively slow performance. This can be improved by replacing natural convection with forced convection. Alternatively, conductive heat transfer can be implemented by using circulating cooling oil or by operating the system in an underwater environment. Another approach is to use multiple smaller SMAs in parallel, increasing the surface-area-to-volume ratio and thereby improving convective heat transfer. All of these methods have been reported in the literature and have proven to enhance the bandwidth of SMAs.

Feasibility Evaluation of SMAs

As stated in section 1.3, the feasibility of using SMAs as actuators in a soft robotic tentacle depends partly on the performance of the controller. Since the PID controller showed consistent behavior and provided a reliable system response, it can be concluded that SMAs are a feasible option for such applications. However, feasibility strongly depends on the specific system requirements. For applications requiring high accuracy and fast response, the use of SMAs remains challenging due to their inherent slow bandwidth and the need for more complex control strategies. In contrast, for systems where high accuracy or speed is less critical, but silent operation and compliance are important, SMAs can offer a viable solution.

Overall, achieving higher performance from the system presented in this research would require significant time and effort. Both the actuator response and the control strategies needed to improve performance are inherently complex. Nevertheless, soft robotic systems should not necessarily be compared directly with high-precision rigid robotic systems, as they are intended for different types of tasks. In applications where compliance, mechanical simplicity, silent operation, or high dexterity are desired, a tentacle actuated by SMAs may provide a simple and effective alternative in the future.

References

[1] K. K. Ahn and N. B. Kha. “Modeling and control of shape memory alloy actuators using Preisach model, genetic algorithm and fuzzy logic”. In: *Mechatronics* 18.3 (2008), pp. 141–152. DOI: 10.1016/j.mechatronics.2007.10.008.

[2] Allegro MicroSystems. *ACS712 Fully Integrated, Hall-Effect-Based Linear Current Sensor IC with 2.4 kVRMS Isolation and a Low-Resistance Current Conductor*. Datasheet, Rev. 22. Allegro MicroSystems. Feb. 2024. URL: https://www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.pdf.

[3] Alpha & Omega Semiconductor. *AOD4184A: 40V N-Channel MOSFET Datasheet*. Rev. 1.0. Alpha & Omega Semiconductor Inc. Sunnyvale, CA, USA, Sept. 2023. URL: https://www.aosmd.com/res/data_sheets/AOD4184A.pdf.

[4] S. Ameduri. “Design of SMA-based structural actuators”. In: *Shape memory alloy engineering: For aerospace, structural, and biomedical applications*. Elsevier, 2021, pp. 485–524. DOI: 10.1016/B978-0-12-819264-1.00014-5.

[5] X. An et al. “Active-Cooling-in-the-Loop Controller Design and Implementation for an SMA-Driven Soft Robotic Tentacle”. In: *IEEE Transactions on Robotics* 39.3 (2023), pp. 2325–2341. DOI: 10.1109/TRO.2023.3234801. URL: https://doi.org/10.1109/TRO.2023.3234801.

[6] Arduino. *Arduino IDE*. https://www.arduino.cc/. Accessed February 2026. 2026.

[7] J. Arias Guadalupe et al. “Efficiency Analysis of SMA-Based Actuators: Possibilities of Configuration According to the Application”. In: *Actuators* 10.3 (2021), p. 63. DOI: 10.3390/act10030063.

[8] Abhishek Chandra Bhatt. *A Brief Analysis of the Hooke’s Joint*. Tech. rep. Technical analysis of universal joint mathematical equations and electro-mechanical compass applications. Diploma 6th sem (Auto), Apr. 2020.

[9] A. Brotzu et al. “Latest attainments”. In: *Shape Memory Alloy Engineering*. Elsevier, 2021, pp. 53–76. DOI: 10.1016/B978-0-12-819264-1.00002-9.

[10] Jeffrey R. Chasnov. *Numerical Methods*. Compiled on 06/09/2025. Hong Kong: LibreTexts™, 2025. URL: https://math.libretexts.org/@go/page/96074.

[11] M. Cianchetti et al. “Bioinspired soft actuation system using shape memory alloys”. In: *Actuators* 3.3 (2014), pp. 226–244. DOI: 10.3390/act3030226.

[12] M. Cianchetti et al. “Design concept and validation of a robotic arm inspired by the octopus”. In: *Materials Science and Engineering: C* 31.6 (2011), pp. 1230–1239. DOI: 10.1016/J.MSEC.2010.12.004.

[13] Deft Dynamics. *Smart Material Actuators*. Accessed: 2026-05-19. Deft Dynamics. 2018. URL: https://deftdynamics.com/shape-memory-alloy-robot-arm/.

[14] J. Z. Ge, L. Chang, and N. O. Pérez-Arancibia. “Preisach-model-based position control of a shape-memory alloy linear actuator in the presence of time-varying stress”. In: *Mechatronics* 73 (2021), p. 102452. DOI: 10.1016/j.mechatronics.2020.102452.

[15] Junghyuk Ko. “Fuzzy PWM-PID Control and Shape Memory Alloy Actuator Design for Cocontracting Antagonistic Muscle Pairs in an Artificial Finger”. Department of Mechanical Engineering. Master of Applied Science. Victoria, BC, Canada: University of Victoria, May 2011.

[16] A. Ktena et al. “A Preisach model identification procedure and simulation of hysteresis in ferromagnets and shape-memory alloys”. In: *Physica B: Condensed Matter* 306.1–4 (2001), pp. 84–90. DOI: 10.1016/S0921-4526(01)00983-8.

[17] X. Li et al. “Disturbance compensation-based output feedback adaptive control for shape memory alloy actuator system”. In: *International Journal of Advanced Robotic Systems* 18.1 (2021). DOI: 10.1177/1729881421993998.

[18] Jhih-Hong Lin and Mao-Hsiung Chiang. “Tracking Control of a Magnetic Shape Memory Actuator Using an Inverse Preisach Model with Modified Fuzzy Sliding Mode Control”. In: *Sensors* 16.9 (2016), p. 1368. DOI: 10.3390/s16091368.

[19] C. Y. Liu and W. H. Liao. “A Snake Robot Using Shape Memory Alloys”. In: *Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Shenyang, China: IEEE, 2004, pp. 601–605. ISBN: 0-7803-8614-8. DOI: 10.1109/ROBIO.2004.1521848.

[20] H. Luo et al. “Hysteresis behaviour and modeling of SMA actuators”. In: *Shape memory alloys*. InTech, 2010, pp. 61–79. DOI: 10.5772/9982.

[21] B. Lynch et al. “Characterization, modeling, and control of Ni-Ti shape memory alloy based on electrical resistance feedback”. In: *Journal of Intelligent Material Systems and Structures* 27.18 (2016), pp. 2489–2507. DOI: 10.1177/1045389X16633764.

[22] M5Stack. *Unit KMeter — K-type Thermocouple Sensor with I²C Interface*. Online documentation. M5Stack. 2026. URL: https://docs.m5stack.com/en/unit/kmeter (visited on 02/25/2026).

[23] S. Majima, K. Kodama, and T. Hasegawa. “Modeling of Shape Memory Alloy Actuator and Tracking Control System with the Model”. In: *IEEE Transactions on Control Systems Technology* 9 (1 2001). PID + feedforward (static hysteresis model).

[24] I. D. Mayergoyz and G. Friedman. “Generalized Preisach model of hysteresis”. In: *IEEE Transactions on Magnetics* 24.1 (1988), pp. 212–217. DOI: 10.1109/20.43892.

[25] Allan Mills. “Robert Hooke’s ’universal joint’ and its application to sundials and the sundial-clock”. In: *Notes and Records of the Royal Society* 61.2 (2007), pp. 219–236. DOI: 10.1098/rsnr.2006.0172.

[26] Nanjing Ouzhuo Technology Co., Ltd. *OH49E Series Linear Hall-Effect IC*. Datasheet. Accessed: 2026-04-28. Nanjing Ouzhuo Technology Co., Ltd., n.d. URL: http://www.ohhallsensor.com.

[27] NASA Glenn Research Center. *Superelastic Tire (LEW-TOPS-99): A non-pneumatic compliant tire utilizing shape memory alloys*. NASA Technology Transfer Portal. Patent Nos. 10,449,804 and 10,427,461; developed as an alternative to pneumatic tires using superelastic shape memory alloys. 2021. URL: https://technology.nasa.gov/patent/LEW-TOPS-99.

[28] ODrive Robotics, Inc. *ODrive S1 Datasheet*. Online Technical Datasheet. Accessed on February 25, 2026. ODrive Robotics, Inc., 2026. URL: https://docs.odriverobotics.com/v/latest/hardware/s1-datasheet.html.

[29] A. Ølander. “An electrochemical investigation of solid cadmium-gold alloys”. In: *Journal of the American Chemical Society* 54.10 (1932), pp. 3819–3833. DOI: 10.1021/ja01349a004.

[30] PJRC. *Teensy development boards*. https://www.pjrc.com/teensy/. Accessed February 2026. 2026.

[31] G. Rizzello and P. Motzki. “Smart materials for mini-actuators”. In: *Endorobotics: Design, R and D and Future Trends*. Elsevier, 2022, pp. 117–163. DOI: 10.1016/B978-0-12-821750-4.00006-2.

[32] Jaroslav Romančík et al. “Shape Memory Alloy Actuators in Robotics”. In: *Actuators* 15.3 (2026), p. 162. ISSN: 2076-0825. DOI: 10.3390/act15030162. URL: https://www.mdpi.com/2076-0825/15/3/162.

[33] Same Sky. *AMT102-V datasheet: AMT10 series modular incremental encoder*. Tech. rep. Rev. 1.15, September 12, 2024. Same Sky, 2024.

[34] S. A. Shabalovskaya. “Electronic structure and properties of NiTi shape memory alloy”. In: *Physica B: Condensed Matter* 312-313 (2002), pp. 1003–1005. DOI: 10.1016/S0921-4526(01)00983-8.

[35] E. Shi et al. “Adaptive control for shape memory alloy actuated systems with applications to human–robot interaction”. In: *Frontiers in Neuroscience* 18 (2024). DOI: 10.3389/fnins.2024.1337580.

[36] Ruben Tolenaars. *Literature review: Artificial muscles for a biomimetic octopus*. Tech. rep. MSc literature review. Delft, The Netherlands: Delft University of Technology, 2025.

[37] D. Trivedi et al. “Soft robotics: Biological inspiration, state of the art, and future research”. In: *Applied Bionics and Biomechanics* 5.3 (2008), pp. 99–117. DOI: 10.1080/11762320802557865.

[38] Á. Villoslada et al. “Position control of a shape memory alloy actuator using a four-term bilinear PID controller”. In: *Sensors and Actuators A: Physical* 236 (2015), pp. 257–272. DOI: 10.1016/j.sna.2015.10.006.

[39] Pauli Virtanen et al. *scipy.optimize.curve_fit — SciPy v1.17.0 Manual*. SciPy Developers. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html.

[40] M. R. Zakerzadeh and H. Sayyaadi. “Precise Position Control of Shape Memory Alloy Actuator Using Inverse Hysteresis Model and Model Reference Adaptive Control System”. In: *Mechatronics* 23 (8 2013). Fig 15: hysteresis model (Prandtl-Ishlinskii) as feedforward term and PI controller as feedback control scheme., pp. 1150–1162. DOI: 10.1016/j.mechatronics.2013.10.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957415813001694.

A

Preisach Update Function Pseudocode

The pseudocode below describes a simplified Preisach update function that does not account for actuator stress. In the full implementation, the Preisach model interpolates between predefined stress levels using bilinear interpolation, as described in section 3.2. For clarity and readability, stress dependence is omitted in the pseudocode presented here.

The function takes as input the current temperature T, the previous temperature T_{previous}, and the maxima and minima lists $alphas$ and $betas$, where $alphas$ is sorted in decreasing order and $betas$ in increasing order. Based on the new temperature T, the function updates the maxima and minima lists, thereby updating the hysteretic state of the actuator, and returns the updated states together with the resulting strain. The function consists of three main parts: first, it determines whether the temperature is increasing or decreasing; second, it updates the maxima and minima lists; and third, it computes the resulting strain.

Algorithm 1 Preisach Model Update Function

function PreisachModel($T, T_{\text{previous}}, alphas, betas$)
Determine sign of temperature derivative
  if $T \geq T_{\text{previous}}$ **then** ▷ heating
    $sign \leftarrow 1$
  else if $T < T_{\text{previous}}$ **then** ▷ cooling
    $sign \leftarrow -1$
  end if

Update extrema lists (wiping-out property)
  if $sign = 1$ **then**
    $\alpha_{\text{new}} \leftarrow T$
    remove from $alphas$ all elements $< \alpha_{\text{new}}$ ▷ wiping-out
    append α_{new} to $alphas$
  end if

  if $sign = -1$ **then**
    $\beta_{\text{new}} \leftarrow T$
    remove from $betas$ all elements $> \beta_{\text{new}}$ ▷ wiping-out
    append β_{new} to $betas$
  end if

Determine strain
  $\epsilon \leftarrow 0$
  $n_{\text{max}} \leftarrow \text{length}(alphas)$

  if $sign = 1$ **then**
    for $k \leftarrow 1$ to $n_{\text{max}} - 1$ **do** ▷ summation over trapezoids and small triangle
      $\epsilon \leftarrow \epsilon + \Big(F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\Big)$
    end for
    $\epsilon \leftarrow \epsilon + F(\alpha_{n_{\text{max}}}, \beta_{n_{\text{max}}-1})$
  end if

  if $sign = -1$ **then**
    for $k \leftarrow 1$ to n_{max} **do** ▷ summation over trapezoids
      $\epsilon \leftarrow \epsilon + \Big(F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\Big)$
    end for
  end if

  $T_{\text{previous}} \leftarrow T$ ▷ update previous temperature

  return $\epsilon, alphas, betas, T_{\text{previous}}$
end function

B

Source Code: Inverse Preisach Model Using the Bisection Method

The following Python function implements the inverse Preisach model used during real-time operation of the Nitinol actuator. The purpose of this function is to determine the required setpoint temperature T_{sp} that results in a reference actuator strain ϵ_r under a given applied stress σ_r. This is done by numerically inverting the forward Preisach model using the bisection method.

Parameters

`stress` **(float)** Reference stress (σ_r) applied to the actuator during real-time operation.

`strain_ref` **(float)** Desired actuator strain (ϵ_r).

`last_T` **(float)** Previously measured actuator temperature, used to determine the correct hysteresis model.

`alphas` **(list)** List of recorded local maxima, storing part of the Preisach state.

`betas` **(list)** List of recorded local minima, storing part of the Preisach state.

`preisach_FORC` **(object)** Dataset containing the FORC surfaces for both heating and cooling models.

`cooling` **(bool)** Boolean indicating the current temperature direction. If `True`, the cooling model is used; otherwise, the heating model is used.

`max_iter` **(int)** Maximum number of iterations allowed in the bisection method to prevent non-terminating searches.

`tol` **(floatl)** Tolerance for the temperature solution. The algorithm terminates when the solution falls within this tolerance.

Description

The function computes the inverse mapping from strain to temperature by repeatedly evaluating the forward Preisach model and adjusting the temperature estimate using the bisection method.

For a given temperature guess, the forward model is evaluated to compute the corresponding strain. The difference between this strain and the desired reference strain defines a root-finding problem. The bisection method is then applied over a bounded temperature interval to locate the temperature at which this difference approaches zero.

To ensure that the search method does not affect the real-time hysteresis state, the Preisach state variables (`alphas` and `betas`) are locally copied and kept constant throughout the search procedure. This guarantees that repeated evaluations of the forward model during the iterative search do not modify the Preisach state.

Returns

`T` **(float)** Estimated setpoint temperature (T_{sp}) that produces the desired strain under the specified stress conditions, within the defined tolerance.

```
def preisach_model_inverse(stress, strain_ref, last_T, alphas, betas, preisach_FORC, cooling,
     max_iter=30, tol=0.01):
    def strain_at(T):
        # Copy mutable inputs: freeze Preisach State
        alphas_local = alphas.copy()
        betas_local = betas.copy()

        if cooling:
            strain, _, _, _ = preisach_model_cooling(stress, T, last_T, alphas_local,
                betas_local, preisach_FORC)
        else:
            strain, _, _, _ = preisach_model_heating(stress, T, last_T, alphas_local,
                betas_local, preisach_FORC)
        return strain

    def bisection_method(f, low, high, tolerance=0.1):
        f_low = f(low)
        f_high = f(high)

        if f_low * f_high > 0:
            return low if abs(f_low) <= abs(f_high) else high

        for _ in range(max_iter):
            if (high - low) / 2 < tolerance:
                return (low + high) / 2

            mid = (low + high) / 2

            if f(mid) == 0:
                return mid
            elif f(low) * f(mid) < 0:
                high = mid
            else:
                low = mid

        return (low + high) / 2

    f = lambda T: strain_at(T) - strain_ref
    T = bisection_method(f, 15, 65, tolerance=tol)

    return T
```

C

PI Temperature Controller Design

This appendix describes the temperature controller used in the PWM-based calibration procedure outlined in section 3.2. More importantly, this controller is also used in the feedforward–feedback (FF–FB) controller described in section 3.5. For simplicity, a PI controller (see Equation C.1) is implemented for both tasks. The controller computes the error $e(t)$ as the difference between the reference temperature and the filtered temperature signal obtained from Equation 2.27. Based on this error, it outputs a duty cycle $u(t)$ that is sent to the PWM module.

The PWM-based calibration procedure requires the SMA to be heated and cooled to predefined temperature levels. Due to the hysteretic behavior of SMAs, the temperature controller must reach these setpoints without overshoot or undershoot, ensuring that the temperature changes monotonically during the experiment. Since the behavior of SMAs depends strongly on whether the temperature is increasing or decreasing—and because the Preisach model relies on the history of local extrema—any over- or undershoot would introduce unintended extrema and lead to incorrect FORC increments. For this reason, the controller gains were experimentally tuned to relatively low values, prioritizing minimal overshoot and undershoot over fast tracking performance. As this approach resulted in stable reference tracking without oscillations, a derivative term was not included. When the temperature controller is used in the cascaded control structure described in section 3.5, the gains are increased, as the objective in that case is faster settling rather than minimizing over- or undershoot.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau \tag{C.1}$$

D

Initial Preisach Temperature-Control-Based Calibration

This appendix presents the calibration method, model results, and failure analysis of the initial temperature-control-based approach, which is similar to the method described by [14]. For clarity, the calibration procedure provided in subsection 3.2.1 is repeated in section D.1. The results of the obtained model and the corresponding failure analysis are provided in section D.2 and section D.3, respectively.

D.1. Preisach Model Identification

As explained in section 2.3 and subsection 3.2.1, the Preisach model is defined by a finite set of FORC-increments that form a FORC surface. These FORC-increments are equally spaced along the Preisach plane and must be empirically identified. By identifying a FORC surface for a selected set of constant stress levels, the Preisach model determines the strain for an intermediate stress level by interpolating between those stress levels. The martensite and austenite saturation temperatures, $\beta_0 = 25°\mathrm{C}$ and $\alpha_0 = 64°\mathrm{C}$, were determined experimentally. The Preisach plane is therefore defined over the triangular domain

$$\{(\alpha, \beta) \mid 25°\mathrm{C} \leq \beta \leq \alpha \leq 64°\mathrm{C}\}.$$

To discretize this domain, both the α- and β-axes were sampled with a uniform step size of $3°\mathrm{C}$. Only grid points satisfying $\alpha > \beta$ were retained, resulting in a triangular mesh over the the Preisach plane. With this discretization, 105 FORC measurements are required to identify a single FORC surface at a constant stress level.

Since the interval between β_0 and α_0 is discretized into 14 equidistant steps of $3°\mathrm{C}$, the identification of a single FORC surface requires 14 experiments. Each experiment corresponds to one reversal temperature α' on the Preisach grid. In each experiment, the following procedure is carried out:

- The actuator is heated from β_0 to a prescribed temperature α' on the discretized α-axis.
- The actuator is then cooled monotonically from α' to β_0.
- For every temperature $\beta' < \alpha'$ on the discretized grid, a FORC increment $F(\alpha', \beta', \sigma)$ is identified.

An example of such an experiment is shown in Figure D.1a, where all FORC increments are plotted for $\alpha' = 64°\mathrm{C}$ at a constant stress level. By performing all 14 experiments, i.e., for each $\alpha' \in [25, 64]°\mathrm{C}$ in steps of $3°\mathrm{C}$, the FORC surface shown in Figure D.1b is obtained. As explained above, this surface defines the Preisach model for a constant stress level. The strain — as given in Equation 2.17 — is calculated using bilinear interpolation whenever a required FORC increment does not lie exactly on a discretization point of the Preisach plane, which is the case in most real-time applications. Bilinear

interpolation is performed by calculating a weighted average of the four nearest neighboring grid points in the two-dimensional (α, β) domain. The interpolated value is computed as

$$f(x,y) \approx f_{00}(1-x)(1-y) + f_{01}x(1-y) + f_{10}(1-x)y + f_{11}xy, \tag{D.1}$$

where f_{00}, f_{01}, f_{10}, and f_{11} denote the FORC increments at the four surrounding grid points of the rectangular cell containing the evaluation point. The normalized local coordinates $x, y \in [0,1]$ are the relative position of the evaluation point within this cell, measured along the α- and β-directions. Therefore, the interpolated value represents the linearly weighted contribution of the four neighboring FORC increments according to the distance of the evaluation point from each grid node.

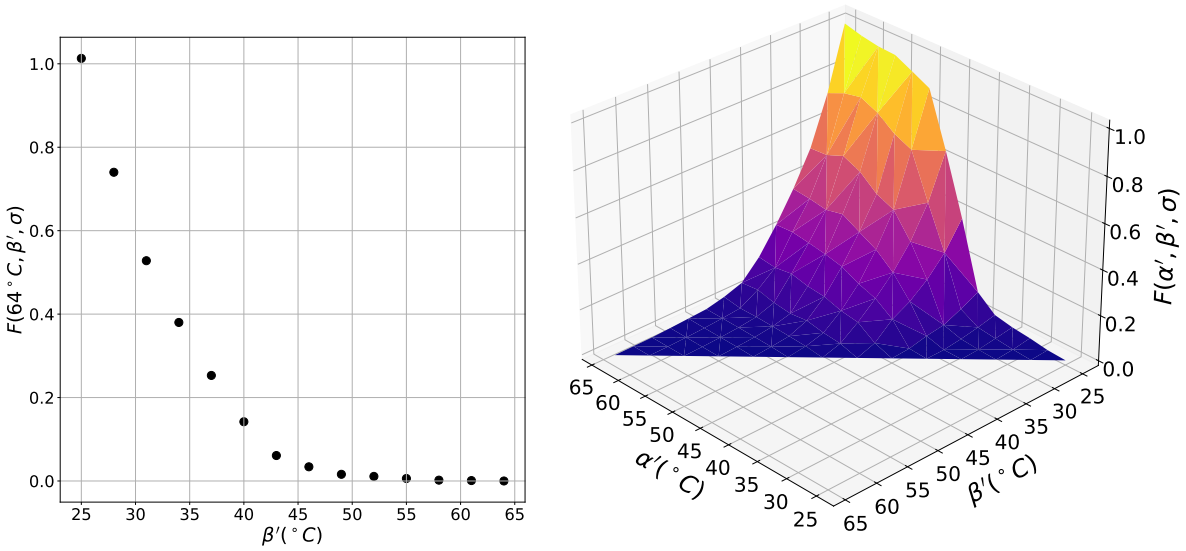


Figure D.1: Examples of FORC increments. (a) Example of a FORC measurement with $\alpha' = 64\,^{\circ}\mathrm{C}$, showing a temperature reversal that yield s 14 FORC increments. (b) Complete set of FORC increments forming the surface that defines the Preisach model.

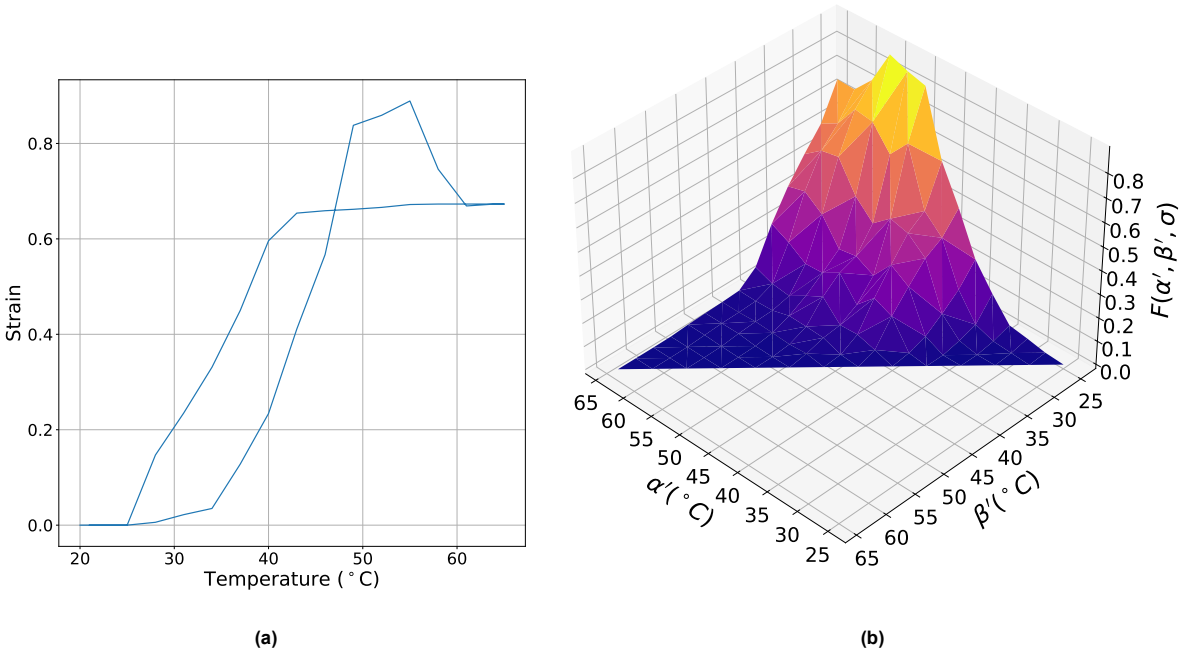
The temperature controller described in Appendix C was used to drive the SMA to the predefined grid points. This process was automated on the microcontroller, which transferred the measured FORC data to Python. A Preisach update function—described in Appendix A—processes the FORC data and computes the strain for a given temperature input, while tracking the recorded maximum and minimum temperatures to maintain the hysteretic state of the actuator.

D.2. Preisach Model Results

The Preisach model obtained after calibration showed disappointing results. An example of the strain–temperature relationship of one of the identified models is shown in Figure D.2a. The plot shows the major strain–temperature hysteresis loop when the temperature is increased from $20\,^{\circ}\mathrm{C}$ to $65\,^{\circ}\mathrm{C}$ (lower curve) and subsequently decreased to $20\,^{\circ}\mathrm{C}$ (upper curve). Rather than the smooth behavior reported in the literature, as illustrated in Figure 2.3 and Figure 2.4, the obtained model shows an oscillatory (bumpy) behavior and, in the provided example, even a cross-over between the heating and cooling curves.

Similarly, the FORC surface shown in Figure D.2b exhibits the same oscillatory behavior, whereas the FORC surfaces reported in the literature [14] appear smooth. Furthermore, the peak of the surface is expected at the reversal point corresponding to the largest temperature drop, namely $F(64\,^{\circ}\mathrm{C}, 25\,^{\circ}\mathrm{C}, \sigma)$, since this is the only instance during the identification process where the temperature is decreased from saturated austenite to saturated martensite. However, as Figure D.2b shows, the largest FORC

increment is obtained at $F(55\,^{\circ}\mathrm{C}, 25\,^{\circ}\mathrm{C}, \sigma)$, where the temperature is decreased not from a saturated austenite state, but from a mixed austenite–martensite state. This unexpected location of the peak explains the cross-over between the heating and cooling curves observed in Figure D.2a.



(a) (b)

Figure D.2: Preisach hysteresis model and corresponding hypersurface obtained from temperature-based calibration under a constant load of $3.0\,\mathrm{N}$. (a) Major strain–temperature hysteresis loop of the Preisach model. The temperature is increased from $20\,^{\circ}\mathrm{C}$ to $65\,^{\circ}\mathrm{C}$ (lower branch) and subsequently decreased back to $20\,^{\circ}\mathrm{C}$ (upper branch). (b) Identified FORC surface defining the calibrated hysteresis model.

Since the literature indicates that the transition between the martensite and austenite states—and therefore the strain under constant load—should occur gradually, the obtained results suggest that the identified model does not represent the true behavior of the actuator. The identification process was repeated multiple times for the same stress level, yielding similarly poor results. Therefore, the identification of the Preisach model for different stress levels was discontinued.

D.3. Analysis of Failure Modes

The poor results described in the previous section were addressed by repeating the identification process with a larger number of samples in the running average temperature filter, thereby reducing the influence of sensor noise on the temperature controller. In addition, the experiment was repeated with different controller gains. Neither approach improved the results of the identified model. As discussed earlier, unexpected FORC increments—and therefore unexpected strain variations—were observed at specific calibration temperatures. Such behavior can occur if the set calibration temperatures differ from the true actuator temperatures, leading to strain discrepancies. Since retuning the controller gains did not improve the results and the controller exhibited good tracking performance without overshoot, it is unlikely that the controller itself caused the poor model identification.

Further analysis of the thermocouple behavior revealed significant drift in the sensor data. Figure D.3 shows the thermocouple measurements, including the running average filter, sampled at 1 Hz while the actuator length and PWM duty cycle were kept constant. Because the electrical input power and the convective heat transfer conditions remain approximately constant under these circumstances, stable and non-oscillatory temperature readings are expected. However, as shown in Figure D.3, the thermocouple output exhibits a slow oscillatory drift. As a result, the temperature controller described in Appendix C was effectively tracking a drifting measurement signal, causing the actual actuator temperature to deviate from the intended calibration temperatures. Consequently, during the identification of

a FORC increment at a given temperature reversal, unintended martensite–austenite phase transformations occurred, leading to incorrect strain differences and thus false FORC increment values.

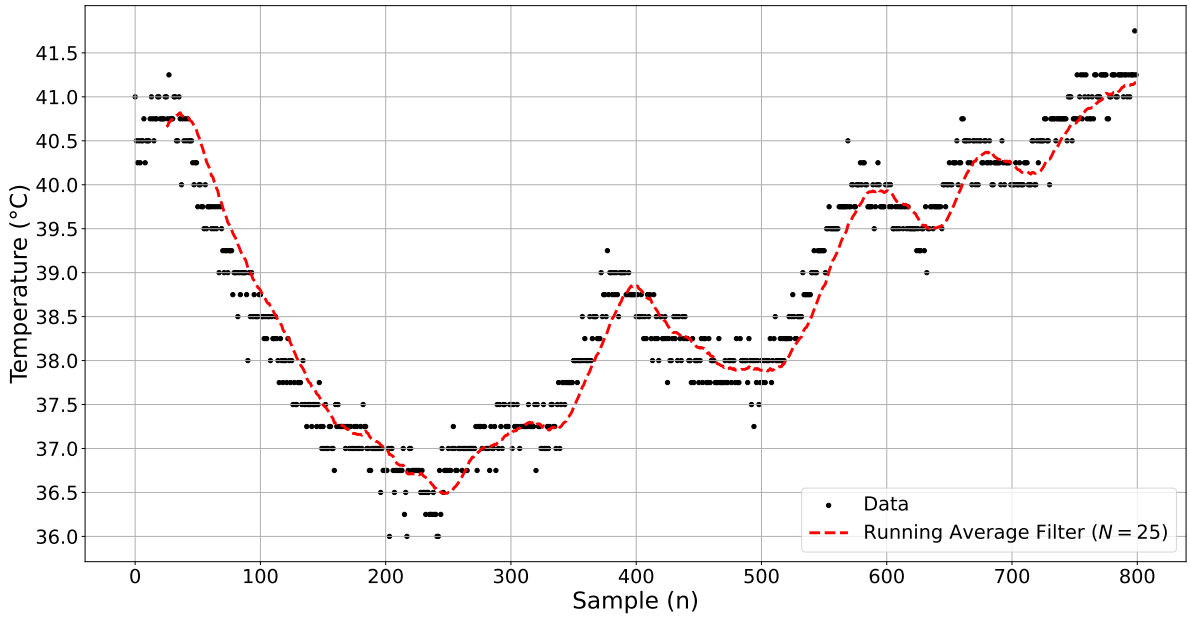


Figure D.3: Thermocouple measurements (black), sampled at 1 Hz, and the corresponding running average over the last 25 samples (red) for a constant temperature. The plot shows significant drift in the temperature readings.

E

Improved Preisach Identification via PWM-Based Actuation

This appendix describes the improved calibration method based using PWM-based actuation, as an alternative to the temperature-control-based approach presented in Appendix D. The motivation for this method is given in section E.1, and the calibration procedure is described in section E.2. The steady-state temperature model used in this approach is explained and validated in section E.3. Finally, the results of the identified Preisach model and their analysis are presented in section E.4 and section E.5.

E.1. Motivation for PWM-Based Identification

As explained in Appendix D, temperature-based calibration of the Preisach model yielded poor results due to drift in the thermocouple readings. This caused the temperature controller to track a temperature different from the intended reference, resulting in faulty FORC increment measurements. This issue could be addressed by installing a different thermocouple or by using multiple thermocouples, as was done in [14]. However, due to the time constraints of this project, new sensors were not implemented, since selecting an appropriate sensor, integrating it into the system, and evaluating its measurement behavior would require much time.

Instead, the calibration procedure can be modified to ensure stable, non-oscillating temperatures during the calibration process. By applying a constant PWM signal to the actuator, the actuator will reach an unknown but stable temperature. Since PWM-based actuation is not part of a closed-loop control scheme—but instead relies on the stable hardware PWM of the microcontroller—the actuator can be heated in such a way that large temperature fluctuations do not occur.

E.2. Experimental Procedure

The problem of PWM-based calibration is that it results in a Preisach model that outputs strain as a function of a steady-state PWM input, rather than temperature. Since the models used in existing control strategies—such as those described in section 2.2—require temperature as an input rather than PWM, a PWM-based Preisach model is unsuitable for control. This is because the formation of martensite and austenite within the SMA is primarily influenced by temperature, and not directly by the current—caused by PWM actuation—flowing through the material.

Therefore, a new calibration method was implemented in which the Preisach model is identified by estimating the temperature at the measured FORC increments during post-processing. Similar to the temperature-based calibration method, the Preisach plane is discretized within a triangular domain. However, instead of equidistant temperature grid points, the domain is defined using equidistant PWM grid points. In the temperature-based approach, martensite and austenite saturation temperatures of $25^{\circ}\mathrm{C}$ and $64^{\circ}\mathrm{C}$ were chosen as the lower and upper bounds of the domain. In the proposed method, duty cycles of 0% and 15% are selected as β_0 and α_0, respectively, since under steady-state condi-

tions these values result in martensite and austenite saturation. Therefore, the Preisach plane can be expressed as the domain of steady-state PWM values

$$\{(\alpha, \beta) \mid 0\% \leq \beta \leq \alpha \leq 15\%\}.$$

Again, this domain is discretized, this time into 20 equidistant steps of 0.75%. The same procedure described in subsection 3.2.1 is used to actuate the SMA to the discretized grid points while measuring the FORC increments. During the experiment, not only the strain (and therefore the FORC increment) is measured, but the temperature, current, and battery voltage are also logged. This sensor data is used during post-processing to estimate the actuator temperature at each FORC increment. In this way, a temperature-based model can be identified rather than a PWM-based model. The temperature estimation method is explained in the next section.

It is important to note that when a certain PWM value is applied, the actuator temperature does not instantly reach its steady-state value but requires time to settle due to thermal lag. Experimentally, it was found that the SMA takes approximately 30 seconds to reach steady-state when transitioning between grid points. Therefore, during the calibration process, the microcontroller sets the desired PWM value, waits for 45 seconds to provide sufficient margin and ensure that steady-state is reached, and subsequently logs the FORC increment together with the sensor data.

E.3. Temperature Estimation via Post-Processing

E.3.1. Temperature Model

To estimate the actuator temperature at the measured FORC increments, a thermal model based on the energy balance of the SMA described in [7] is used and defined as

$$\rho C V \frac{dT}{dt} = Q_s + Q_{\text{latent}} - Q_{\text{convection}} - Q_{\text{conduction}} - Q_{\text{radiation}}, \tag{E.1}$$

where ρ, C, and V denote the density, specific heat, and volume of the SMA, respectively, and T is the actuator temperature. The term Q_s represents the heat generated through Joule heating, Q_{latent} the heat generated by the solid-state phase transformation between martensite and austenite, $Q_{\text{convection}}$ the heat transfer to the surrounding air, $Q_{\text{conduction}}$ the conductive heat transfer within the material, and $Q_{\text{radiation}}$ the radiative heat transfer.

According to [7], the thermal behavior of SMA actuators is primarily governed by Joule heating, latent heat transfer during phase transformation, and convective heat transfer. Conduction is neglected due to the low internal thermal resistance of SMAs, and it is assumed that the temperature distribution within the actuator is uniform. Furthermore, because radiative heat transfer is proportional to T^4, it is neglected at the relatively low operating temperatures of the actuator.

The convective heat transfer can be expressed as

$$Q_{\text{convection}} = hA(T - T_e), \tag{E.2}$$

where h is the convective heat transfer coefficient, A the surface area of the SMA, and T_e the environmental temperature. The latent heat transfer is expressed as

$$Q_{\text{latent}} = \rho V \Delta H \frac{d\xi}{dt}, \tag{E.3}$$

where ΔH is the specific latent heat of transformation and ξ is the martensite volume fraction. The heat generated by Joule heating can be expressed as

$$Q_s = P_e = U \cdot I_{\text{average}}, \tag{E.4}$$

where P_e is the electrical power supplied to the actuator, defined as the product of the voltage U and the average current I_{average}, obtained from the filtered current sensor measurements. As explained in subsection 2.4.2, the PWM module is operated at a low switching frequency of 1000 Hz, resulting in negligible switching losses and therefore a negligible voltage drop across the module. Consequently, the measured battery voltage of the test bench is used as an estimate for U.

Substituting Equation E.2, Equation E.3, and Equation E.4 into Equation E.1 yields the energy balance for the Nitinol actuator

$$\rho C V \frac{dT}{dt} = U \cdot I_{\text{average}} + \rho V \Delta H \frac{d\xi}{dt} - hA(T - T_e). \tag{E.5}$$

Since the measurements are recorded under steady-state conditions, the time-derivative terms vanish. Therefore, the steady-state energy balance reduces to

$$U \cdot I_{\text{average}} = hA(T - T_e). \tag{E.6}$$

Since the actuator surface area A increases with its length L, while the convective heat transfer coefficient h is assumed to be constant, the product hA is approximated as a linear function of the actuator length,

$$hA(L) = aL + b, \tag{E.7}$$

where a represents the slope of the linear function and b denotes an offset. Substituting this linear expression into the steady-state energy balance yields an estimate for the actuator temperature

$$T = T_e + \frac{U \cdot I_{\text{average}}}{aL + b}, \tag{E.8}$$

where a and b must be determined experimentally. Since the Preisach plane is discretized into 231 grid points, these constants can be estimated by fitting the model to the 231 recorded data points. A least-squares optimization algorithm from SciPy [39] is used to compute a and b for each identified FORC surface during post-processing. Subsequently, using the encoder readings, filtered current measurements, and filtered battery voltage measurements, the actuator temperature at each FORC increment is estimated using Equation E.8. The source code of the temperature model is provided in Appendix F.

It is important to note that the actuator temperature estimate provided by Equation E.8 is a simplification of the true actuator temperature. Several sources of error can therefore be expected, originating from:

- Measurement errors in the current sensor, battery voltage sensor, and encoder readings.
- The assumption that the convective heat transfer term hA can be approximated as a linear function of the actuator length L.
- The assumption that conductive heat transfer is negligible. In practice, the heat distribution along the SMA is not perfectly uniform, and conductive heat transfer occurs at both ends of the actuator where it is clamped to the electrical connections.
- The assumption that radiative heat transfer is negligible.
- Power losses in the PWM module (although, as discussed previously, these are expected to be small).

E.3.2. Temperature Model Results

As discussed, the temperature model defined in Equation E.8 is used during post-processing to estimate the actuator temperature at the measured FORC increments. Four individual FORC surfaces are identified for four different load cases (1.5 N, 2.0 N, 2.5 N, and 3.0 N). Consequently, four separate temperature models are fitted, one for each FORC surface identification.

The temperature model and filtered temperature sensor data for the load case of 3.0 N are shown from two different viewpoints in Figure E.1. The transparent surface represents the temperature model defined in Equation E.8, while the dots denote the filtered temperature readings logged during the experiment. The measurements follow the trend defined by the surface, as most data points lie close to the model surface. However, due to the aforementioned drift present in the thermocouple, it is difficult to determine how accurately the temperature model represents the true actuator temperature. The length–power–temperature plots for the other load cases are presented in Appendix F.

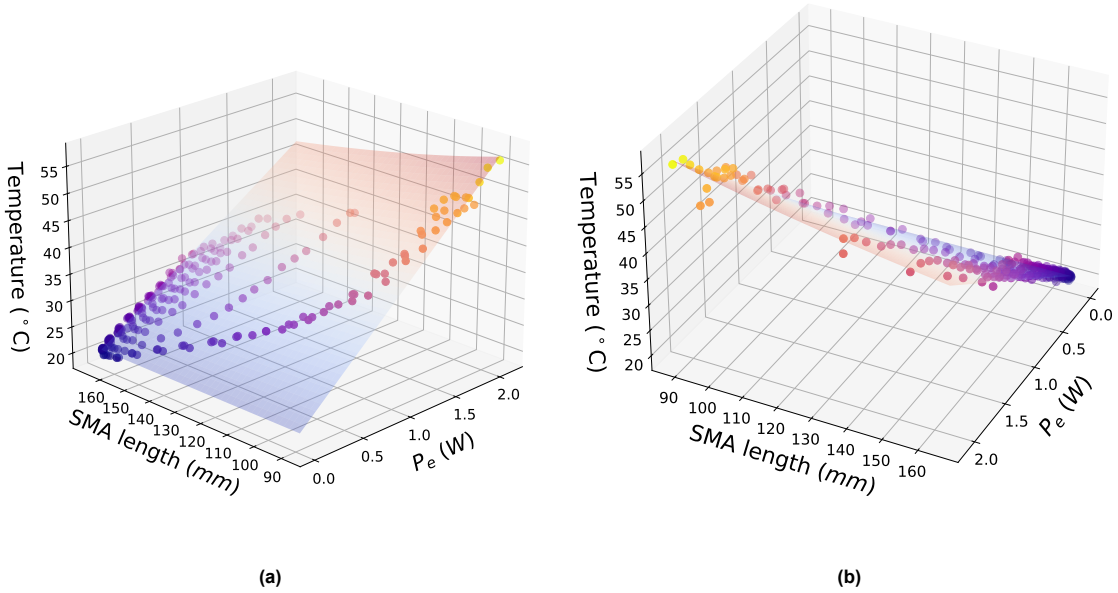


Figure E.1: Two views of the temperature model, plotted as a surface, together with the temperature measurements, plotted as dots. The temperature model is based on the steady-state energy balance defined in Equation E.8, where the convective heat transfer term hA is approximated as a linear function of the actuator length L (Equation E.7). The model parameters a and b were obtained through least-squares optimization using the 231 discretized grid points of the Preisach plane.

E.4. Preisach Model Results

Compared to the temperature-based calibration results presented in Appendix D, the PWM-based Preisach model demonstrates more consistent and physically plausible behavior. The model was identified for four actuator loads: 1.5 N, 2.0 N, 2.5 N, and 3.0 N. The four major hysteresis loops, one for each applied load, are shown in Figure E.2. Each loop represents the Preisach model response as the temperature increases from 20 °C to 60 °C and then decreases back to 20 °C. The figure shows consistent results across the different load levels and, unlike the temperature-based calibration, exhibits heating and cooling curves without crossovers.

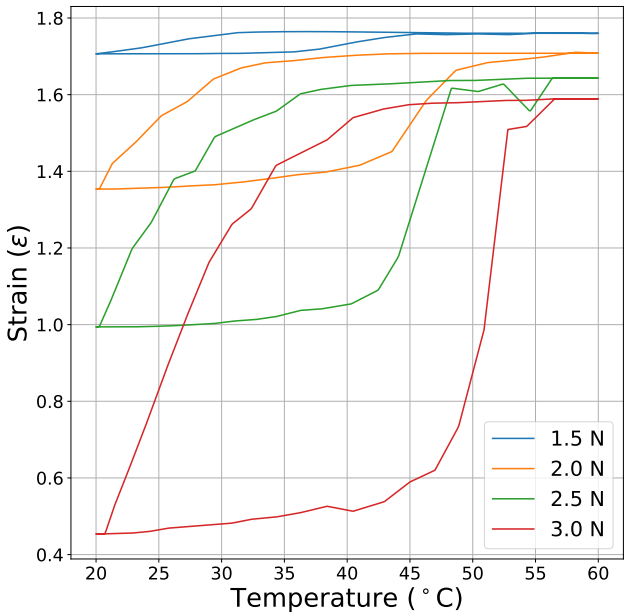
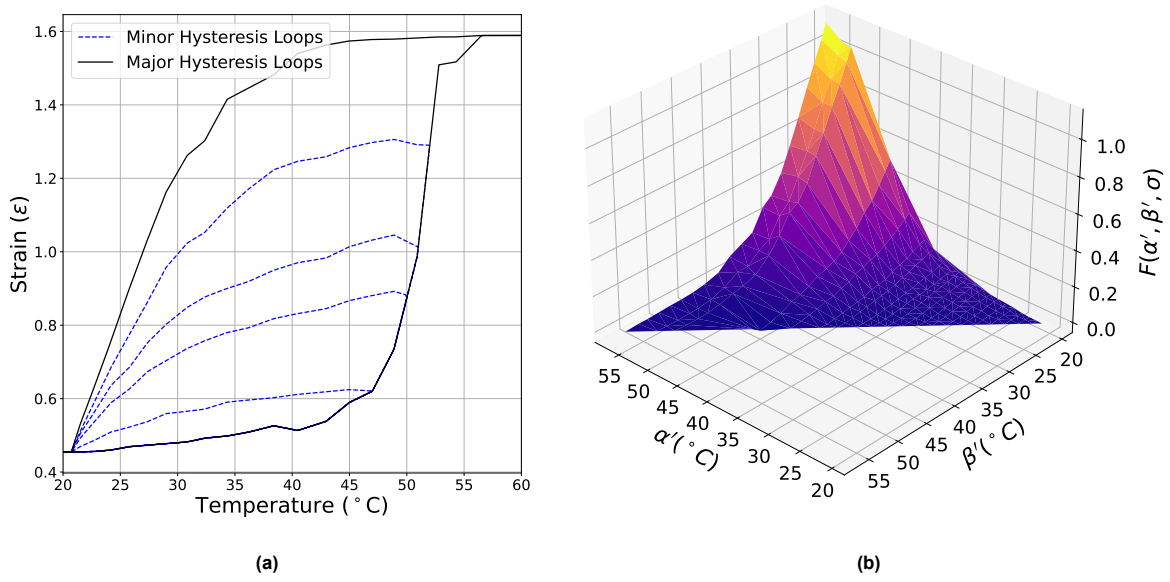


Figure E.2: Temperature–strain major hysteresis loops of the Preisach model identified using PWM-based actuation for four applied loads: 1.5 N, 2.0 N, 2.5 N, and 3.0 N. The loops show the Preisach model response during heating from 20 °C to 60 °C and subsequent cooling back to 20 °C.

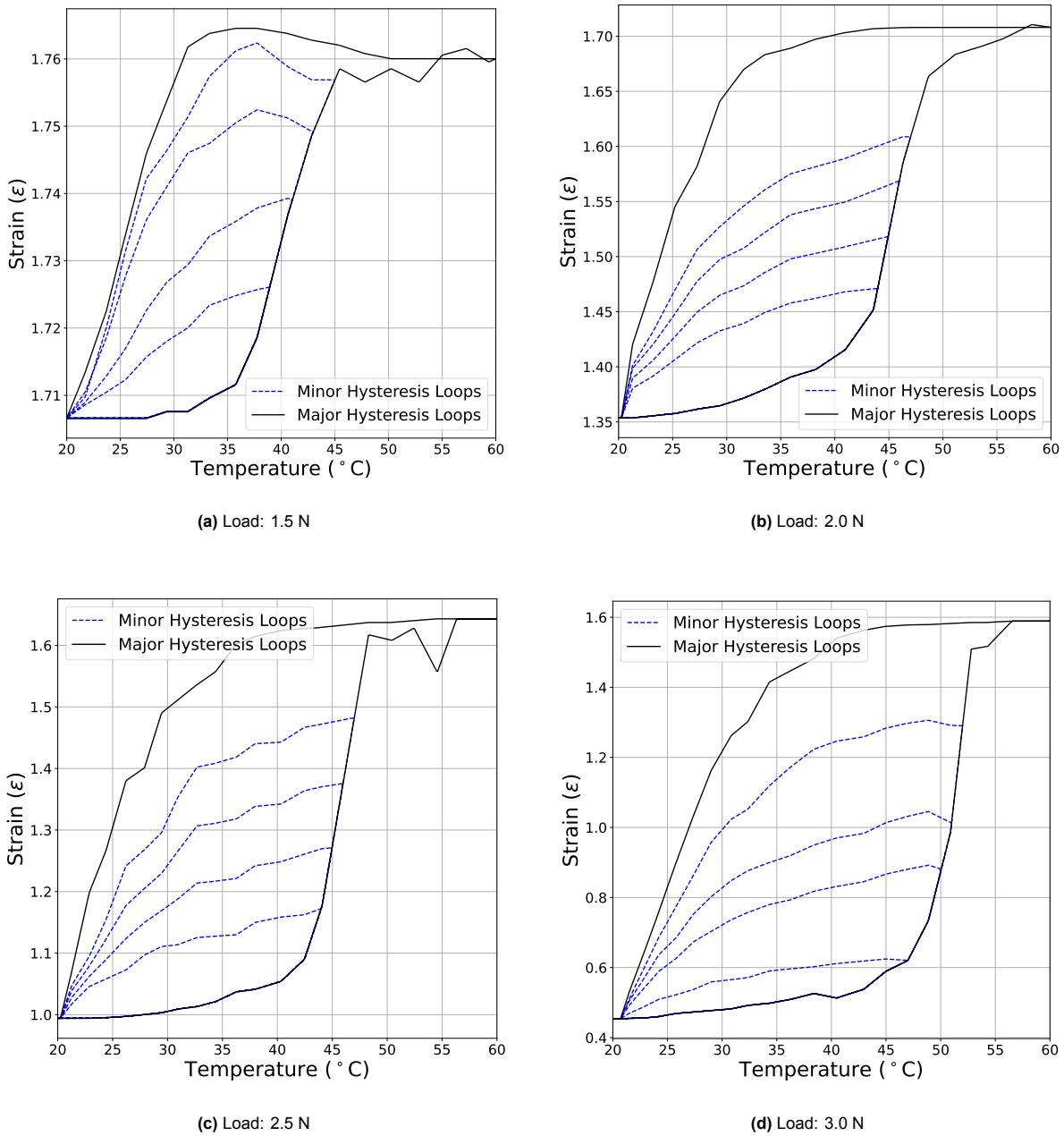
The major hysteresis loop for a 3.0 N load from Figure E.2 is shown in Figure E.3a, now including several minor hysteresis loops. In these minor loops, the temperature is increased from 20 °C to a point where the actuator is in a hybrid martensite–austenite state, and then returned to 20 °C. Additionally, Figure E.3b presents the corresponding FORC surface. Upon visual inspection, this surface is noticeably smoother and free of unexpected peak locations compared to those obtained during temperature-based calibration.



(a) **(b)**

Figure E.3: Results of the identified Preisach model for a 3.0 N load. (a) Temperature–strain relationship showing major and minor hysteresis loops. (b) Corresponding identified FORC surface.

The major hysteresis loop (black) for each applied load, together with several minor hysteresis loops (blue), is shown in Figure E.4. The minor loops represent the model output when the temperature is increased to a specified value and subsequently decreased back to 20 °C.



(a) Load: 1.5 N

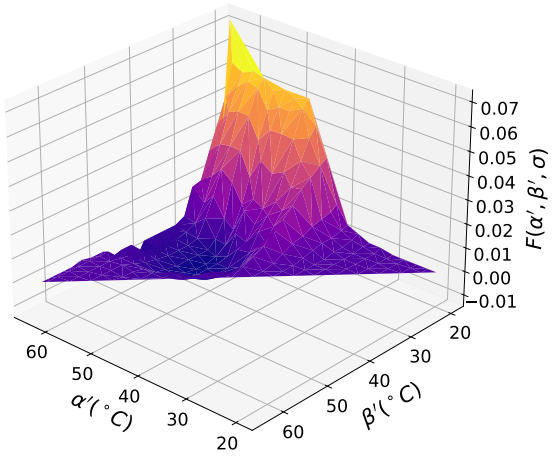
(b) Load: 2.0 N

(c) Load: 2.5 N

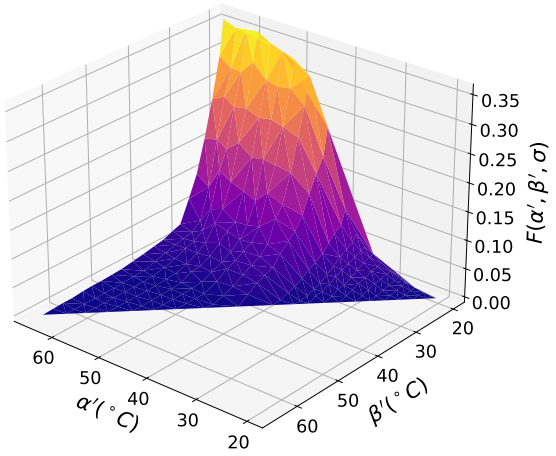
(d) Load: 3.0 N

Figure E.4: Temperature–strain major hysteresis loop (black) and various minor hysteresis loops (blue) of the Preisach model identified using PWM-based actuation. Subfigures (a), (b), (c), and (d) correspond to applied loads of 1.5 N, 2.0 N, 2.5 N, and 3.0 N, respectively.

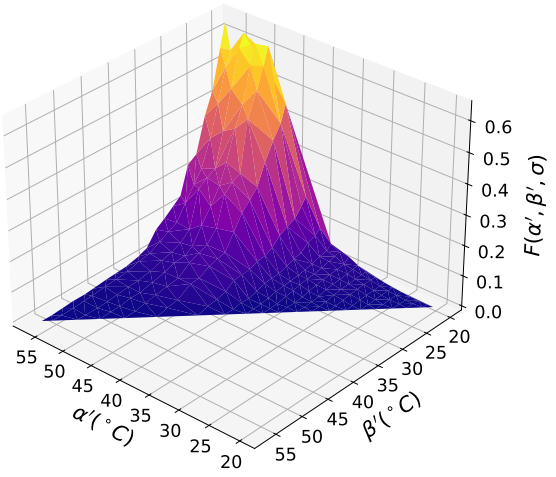
The identified FORC surfaces for the individual load cases are presented in Figure E.5.



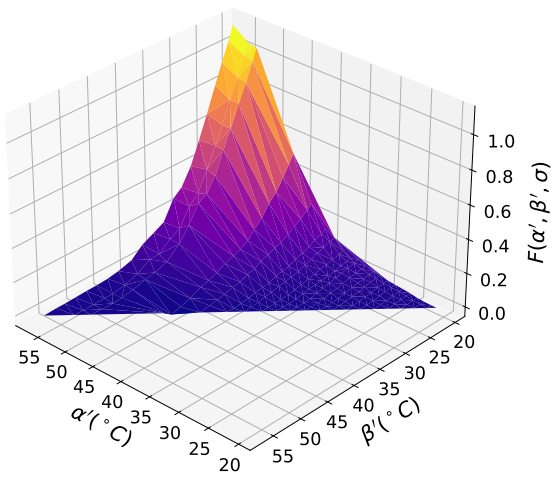
(a) Load: 1.5 N



(b) Load: 2.0 N



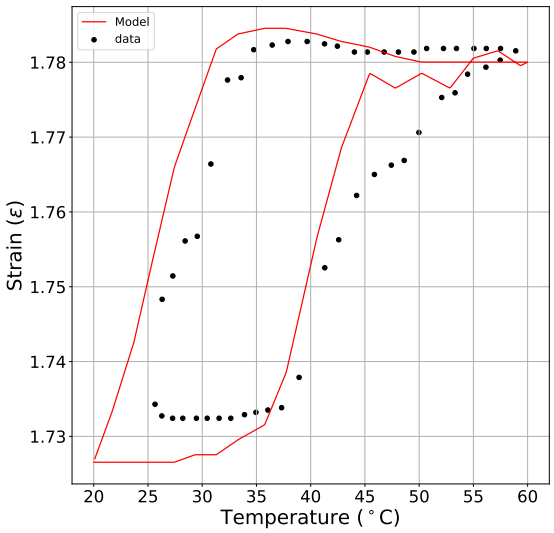
(c) Load: 2.5 N



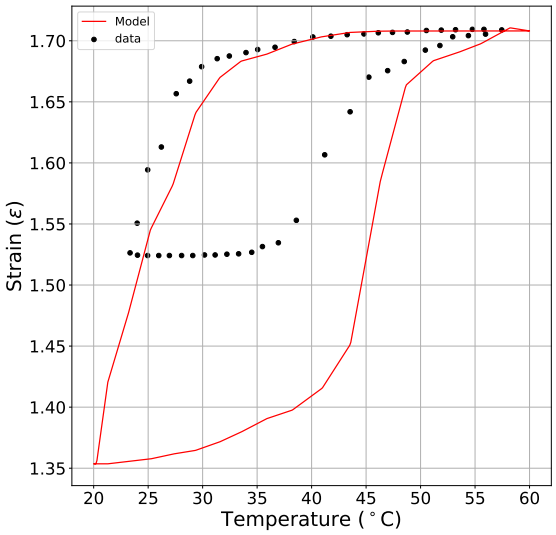
(d) Load: 3.0 N

Figure E.5: Identified FORC surfaces of the Preisach model obtained using PWM-based actuation and the temperature estimation method described in section E.3. Subfigures (a), (b), (c), and (d) show the surfaces corresponding to loads of 1.5 N, 2.0 N, 2.5 N, and 3.0 N, respectively.

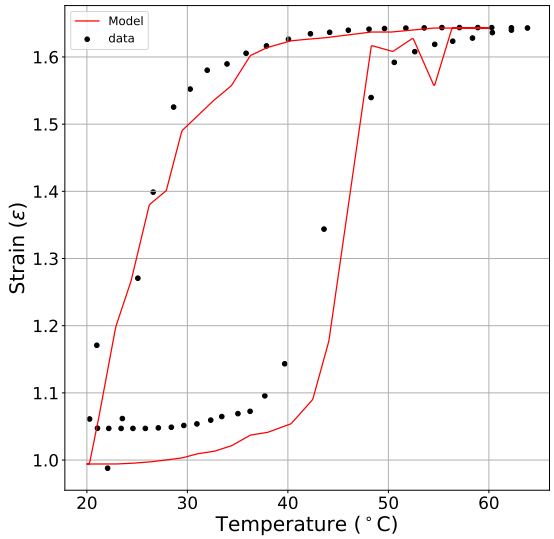
The figures below show the comparison between the identified model predictions and experimental data. The experimental data were obtained by recording strain during PWM-based actuation. In post-processing, the temperature of the experimental data was estimated using the same method applied in model identification. As a result, the temperature–strain data include temperature estimates based on current, voltage, and encoder readings.



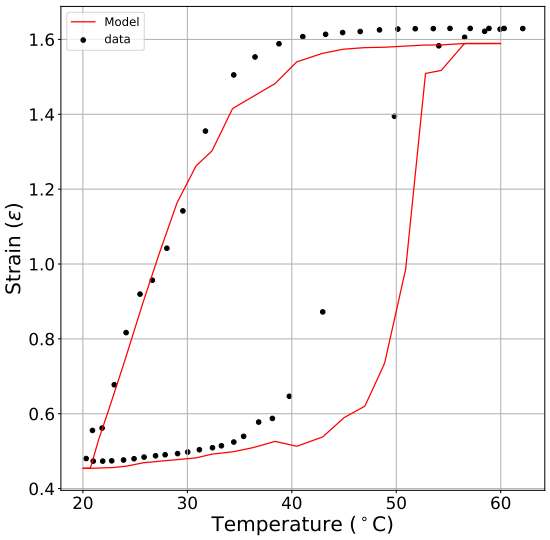
(a) Load: 1.5 N



(b) Load: 2.0 N



(c) Load: 2.5 N



(d) Load: 3.0 N

Figure E.6: Comparison of the major hysteresis loop of the identified Preisach model with experimental data. For the experimental data, strain was measured by heating the actuator from saturated martensite to saturated austenite and back to saturated martensite using PWM actuation. In post-processing, the temperature for each strain data point was estimated using the method described in section E.3. Subfigures (a), (b), (c), and (d) correspond to applied loads of 1.5 N, 2.0 N, 2.5 N, and 3.0 N, respectively.

E.5. Model Evaluation

E.5.1. Evaluation Approach

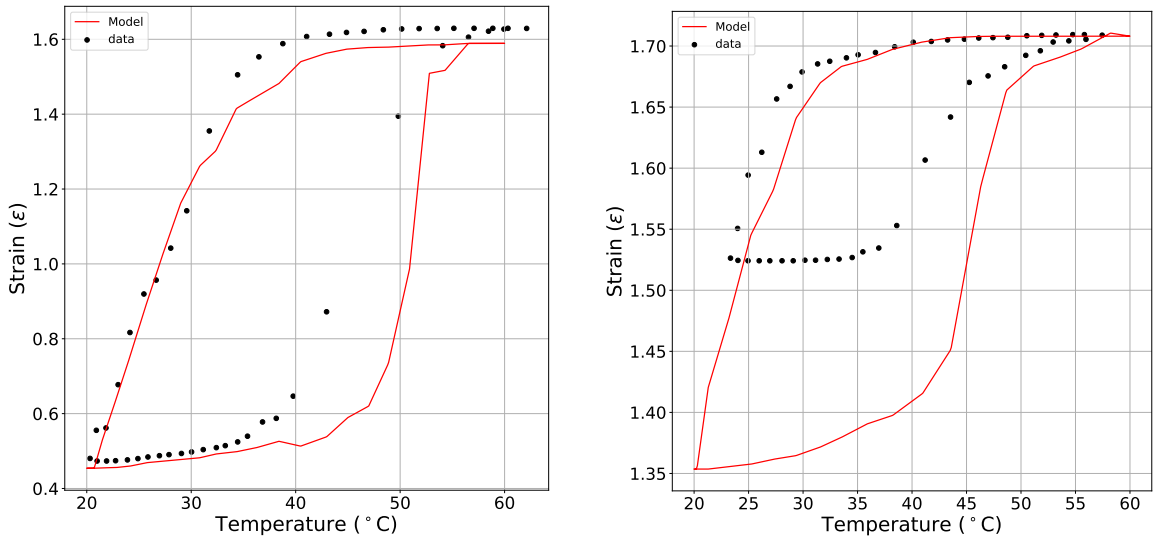
To verify the identified Preisach model, a separate experiment was performed for each load case in which the temperature–strain relationship was recorded during a full heating–cooling cycle, i.e., a major hysteresis loop. In these experiments, the PWM duty cycle was increased in equal steps from 0%—where the actuator is in saturated martensite—to 15%—where the actuator is in saturated austenite—and subsequently decreased in equal steps back to 0%. As in the FORC identification experiments, the current, battery voltage, and encoder readings were logged in order to estimate the actuator temperature using the model given in Equation E.8. The estimated temperature was then used to construct the corresponding temperature–strain relationship.

Direct temperature measurements from the thermocouple were not used as a reference in this validation. As discussed, the thermocouple outputs significant drift during the experiments, which prevents it from serving as a reliable representation of the true actuator temperature. Using these measurements directly would therefore introduce errors in the temperature–strain relationship and could lead to misleading conclusions about model performance. Therefore, the validation is performed using the same temperature estimation method that was employed during the FORC identification procedure. This ensures consistency between the identification and validation datasets, but it also implies that the comparison is made using estimated rather than directly measured temperatures.

It should therefore be noted that this validation assesses the predictive performance of the combined thermal and hysteresis model. Any systematic error in the temperature estimation model cannot be detected in this procedure, meaning that the hysteresis model and temperature estimator are effectively validated together rather than the Preisach model being validated independently.

E.5.2. Low Performance of Heating Trajectories

A comparison between the model and the experimental data for an applied load of 3.0 N is shown in Figure E.7a. The figure shows that the model captures the overall trend of the hysteretic temperature–strain relationship reasonably well. However, a noticeable strain difference is present in the heating curve between approximately $40\,^{\circ}$C and $50\,^{\circ}$C, where the predicted strain deviates from the experimental data. Two possible explanations can be identified for this difference. First, inaccuracies in the estimated temperature may introduce errors in the temperature–strain relationship. Second, the data used for the FORC identification was recorded during temperature reversals, i.e., during cooling, whereas the experimental data shown in Figure E.7 was obtained during heating. Therefore, since the model was calibrated using cooling data, it may not accurately capture the behavior of the SMA during heating. To verify this, a FORC identification method in which the model is identified during heating is required.



(a) Applied load of 3.0 N.

(b) Applied load of 2.0 N.

Figure E.7: Comparison between the Preisach model and experimentally measured temperature–strain relationships during a full heating–cooling cycle (major hysteresis loop) for two different applied loads. The PWM duty cycle was increased from 0% to 15% and subsequently decreased back to 0%. The model captures the general hysteretic behavior of the actuator for both loads, although differences between the model and experimental data are visible.

E.5.3. Irregularities in Model Behavior

Besides differences between heating and cooling performance, the model shows small irregularities or ”bumps” in the temperature–strain relationship. These could be caused by friction in the test bench. In particular, friction in the pink encoder wheel bearings described in section 2.4, as well as friction inside the encoder itself, may introduce small inconsistencies in the measured displacement. The encoder does not contain internal bearings, and variations in friction can be felt when manually rotating the encoder wheel, which supports this explanation. Moreover, the non-smooth trajectories of the model could be the result of errors in the previously discussed temperature estimation approach.

For lower applied loads, as shown in Figure E.7b, the differences between the model and the experimental data become more pronounced. One possible reason is again the inconsistent friction in the encoder, which may have a larger relative influence at lower loads. Another contributing factor may be the difference in experimental duration. The FORC identification experiments took approximately twice as long as the validation experiments. It was observed that the actuator slowly elongates in its martensitic state under a static load at low temperatures. As a result, the model appears as a ”stretched” version of the experimental curve. It should be noted, however, that the difference in strain is somewhat exaggerated in Figure E.7b, as the figure is shown on a more zoomed-in scale than Figure E.7a.

E.5.4. Main Sources of Error and Improvements

To summarize the model behavior, the model shows good agreement with the experimental data during cooling, but poor agreement during heating. One possible explanation is that the FORC identification was performed using cooling data, which may prevent the model from accurately capturing the heating behavior. To address this issue, two FORC surfaces could be identified for each stress level: one for cooling and one for heating. With this approach, the model output is computed using one of the two FORC surfaces, depending on the direction of the temperature change.

Moreover, an important observation was made during the testing of the identified models: the previously discussed drift in the temperature sensor occurs only during heating. When the PWM module is switched off, the drift disappears. Therefore, the observed ”bumpy” behavior could be addressed by abandoning the temperature estimation approach and simplifying the FORC identification procedure. By directly logging the temperature and strain during temperature reversals, multiple FORC increments

can be identified by fitting curves to the obtained temperature–strain data. In other words, this can be achieved by fitting curves to the data shown in Figure D.1a.

Therefore, subsection 3.2.2 presents a third calibration approach in which the model is calibrated twice for each stress level: once during heating and once during cooling. In addition, the FORC identification method is simplified by identifying the FORC increments through curve fitting of the temperature reversals.

F

Temperature Estimation: Implementation and Results

This appendix provides the source code used for estimating the actuator temperature during the FORC experiments and the resulting fitted data across the four load cases.

F.1. Source Code

The following python function creates a processed DataFrame from the measurement data obtained during the FORC experiments. It augments the raw data with calculated strain and temperature estimates.

Parameters

`original_df` **(pandas.DataFrame)** Contains preprocessed measurement data, including:

- Filtered current and temperature measurements.
- Actuator position (derived from the calibrated encoder position).
- Filtered battery voltage readings (`Vbat`).

`Te` **(float)** The ambient or baseline temperature used for the estimation model.

Description

The function produces a processed dataset by calculating the SMA strain from actuator position data and generating temperature estimates via a convection-based model ($T = T_e + \frac{P_e}{aL+b}$). The convection parameters (a and b) are fitted using `scipy.optimize.curve_fit`.

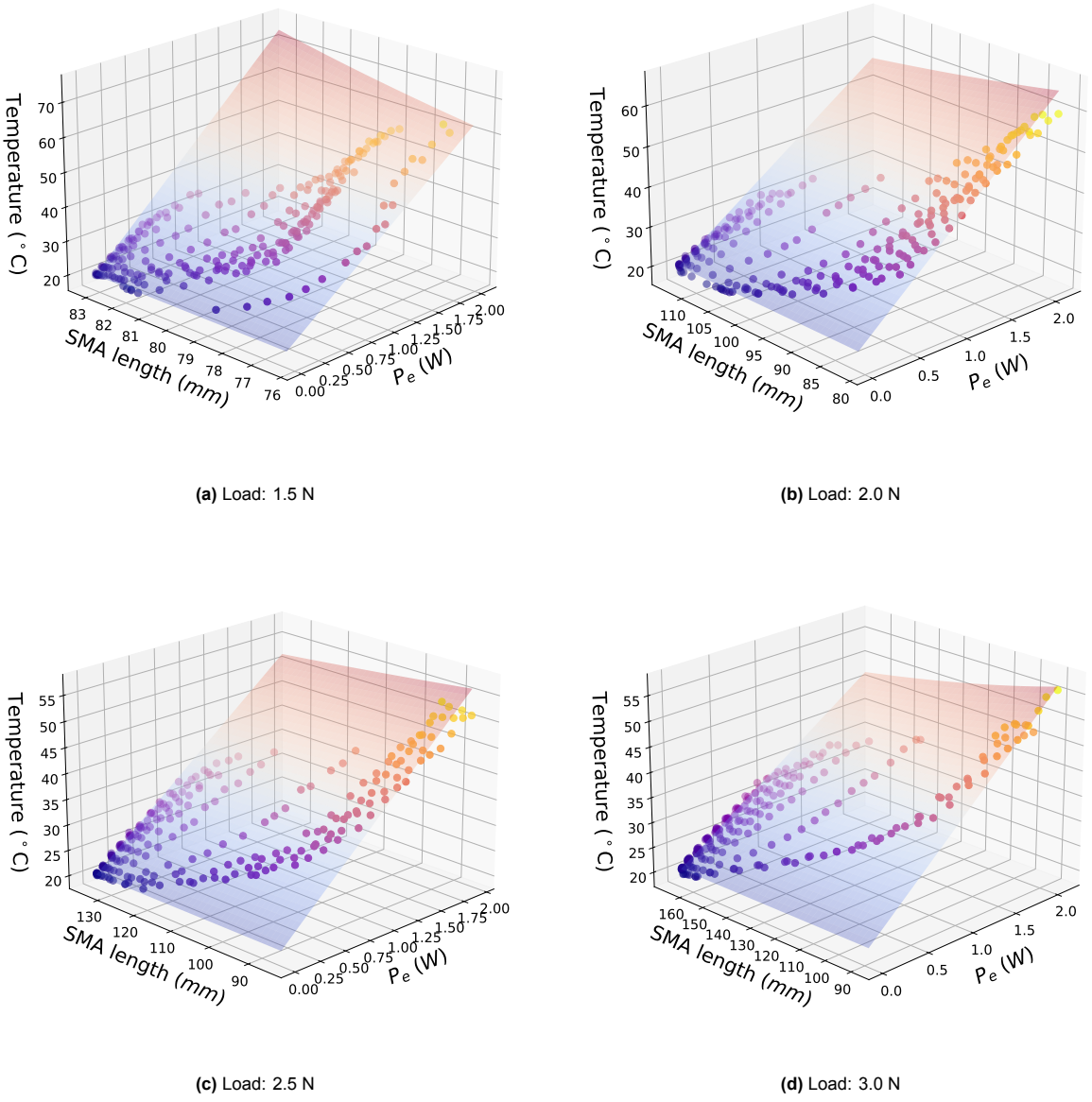
Returns

`processed_df` **(pandas.DataFrame)** A new DataFrame containing the original measurements plus the newly calculated `temp_fit` and `strain` columns.

```python
import numpy as np
from scipy.optimize import curve_fit
import pandas as pd

def create_processed_dataset(original_df, Te):
    # Compute strain from position
    pos = original_df["pos"].astype(float)
    strain = (max_extention - pos) / L0

    # Copy the dataframe and add strain column
    processed_df = original_df.copy()
    processed_df["strain"] = strain

    # Obtain current, Vbat, power, temperature and SMA length arrays
    current_array = original_df["current"].to_numpy()
    Vbat_array = original_df["Vbat"].to_numpy()
    power_array = current_array * Vbat_array
    temp_array = original_df["temperature"].to_numpy()
    length_array = original_df["pos"].to_numpy() + L0

    # Define temperature model: T = Te + P / (a*L + b)
    def temperature_model(vars, a, b):
        P, L = vars
        return Te + P / (a * L + b)

    # Initial guesses for a and b
    p0 = [0.01, 1.0]

    # Fit the convection parameters
    params, cov = curve_fit(
        f=temperature_model,
        xdata=(power_array, length_array),
        ydata=temp_array,
        p0=p0)

    a_fit, b_fit = params

    # Add fitted temperature column to processed dataframe
    temp_fit = temperature_model((power_array, length_array), a_fit, b_fit)
    processed_df["temp_fit"] = temp_fit

    # Return the processed dataframe
    return processed_df
```

F.2. Estimation Results

The results of the fitting procedure for each identified FORC surface identified using the PWM-based calibration approach are visualized in Figure F.1.



(a) Load: 1.5 N

(b) Load: 2.0 N

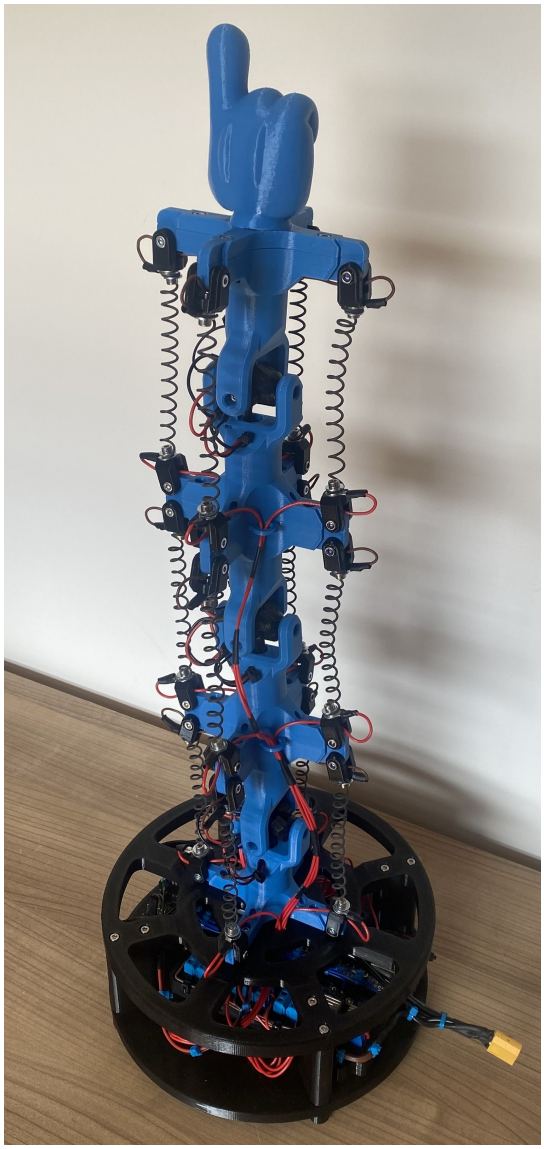
(c) Load: 2.5 N

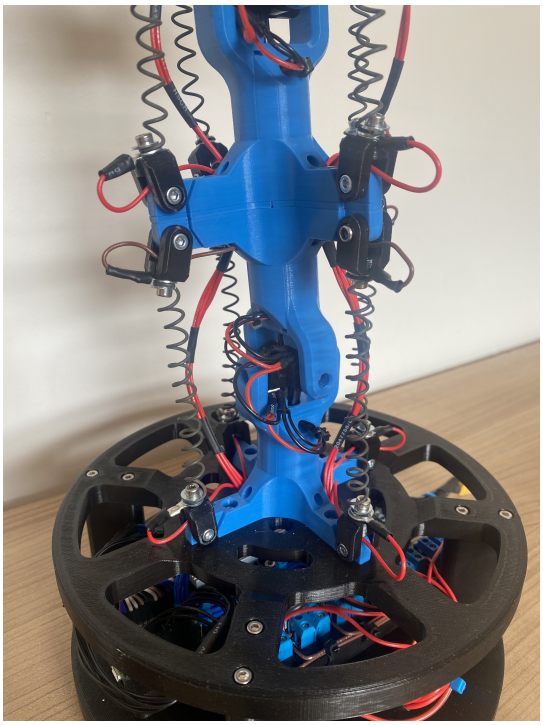
(d) Load: 3.0 N

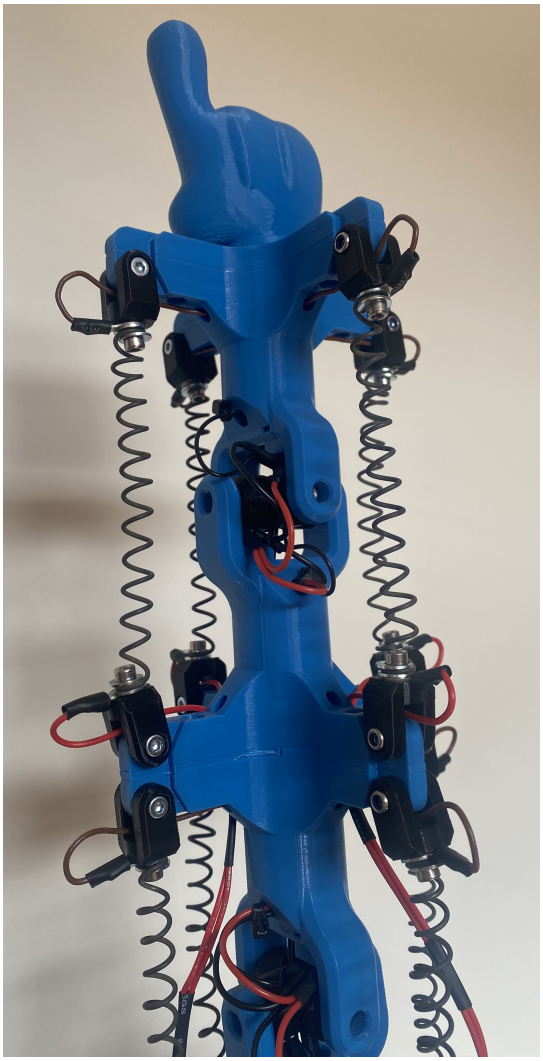
Figure F.1: Estimated actuator temperature T across the identified FORC surfaces for different mechanical loads. The temperature is calculated using the steady-state energy balance model defined in Equation E.8, where the convective heat transfer hA is treated as a linear function of the actuator length L (Equation E.7). The model parameters a and b were obtained through least-squares optimization against the 231 discretized grid points of the Preisach plane.

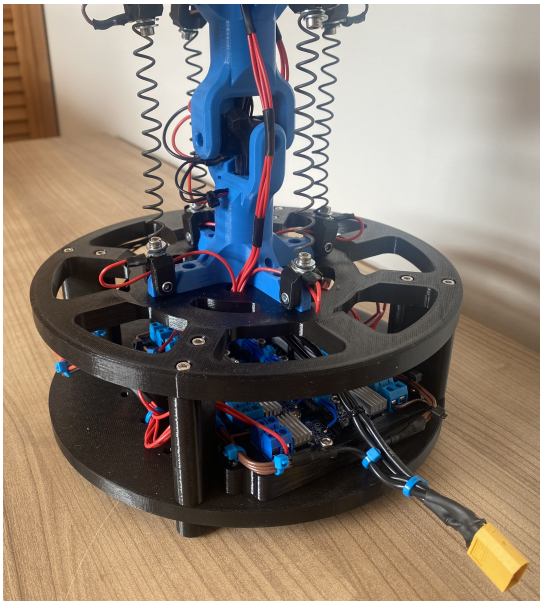
G

Images of the Fabricated Tentacle







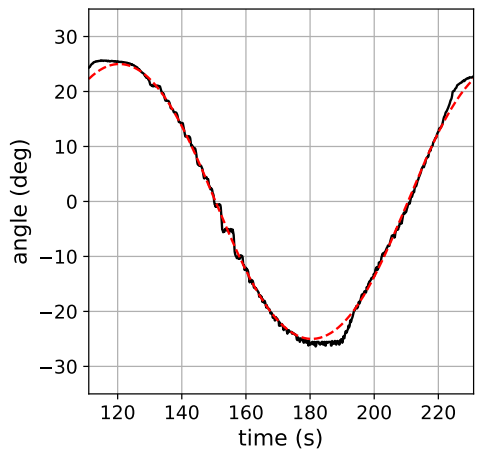


H

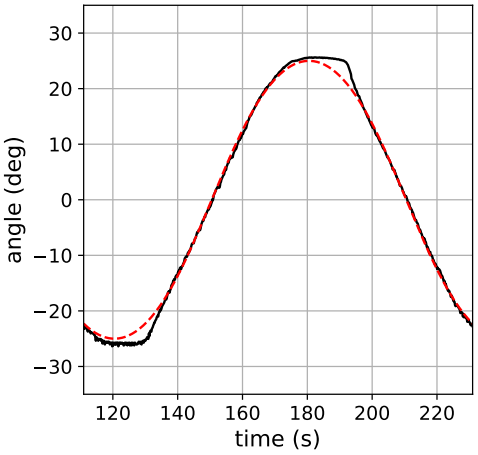
Sinusoidal Reference Tracking Results

This appendix provides the results of the experiment described in section 4.3, where the designed tentacle tracks a sinusoidal reference. The time–angle response of all axes are provided in section H.1, section H.2 and section H.3, corresponding to reference frequencies of $\frac{1}{120}$ Hz, $\frac{1}{60}$ Hz and $\frac{1}{30}$ Hz, respectively.

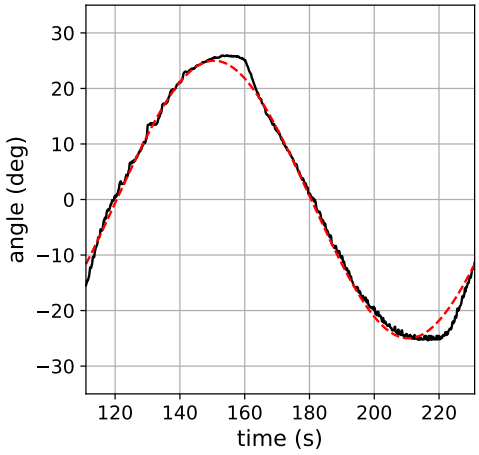
H.1. Response at $\frac{1}{120}$ Hz



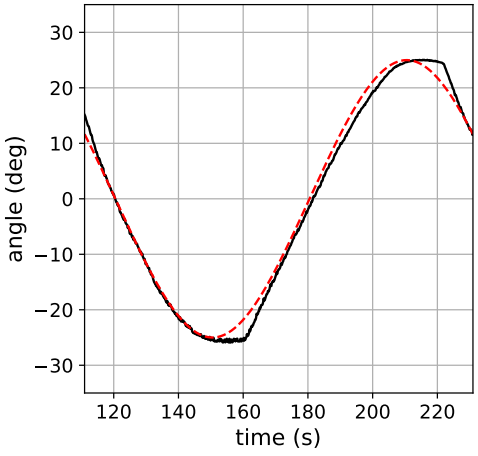
(a) x_1 response



(b) x_2 response



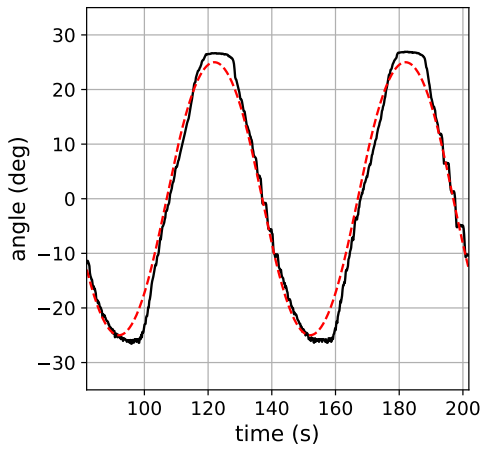
(c) y_1 response



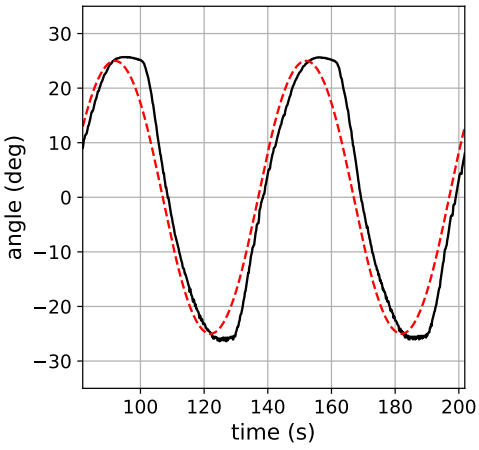
(d) y_2 response

Figure H.1: Response of all axes to a sinusoidal reference input at $\frac{1}{120}$ Hz. The reference signal is shown in red, while the black curve denotes the filtered Hall sensor measurement.

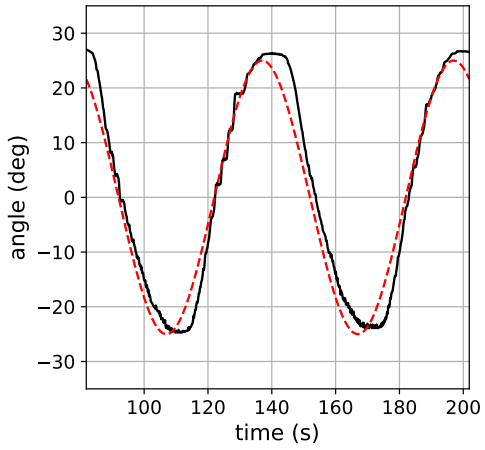
H.2. Response at $\frac{1}{60}$ Hz



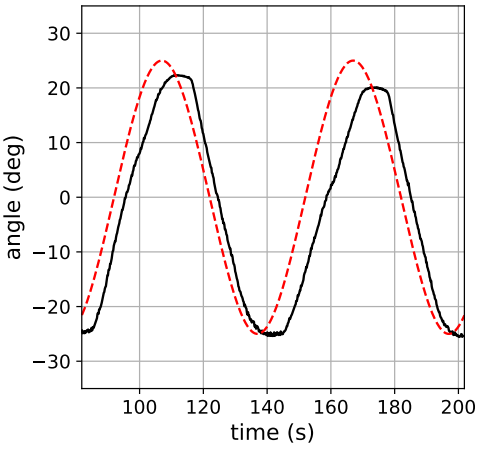
(a) x_1 response



(b) x_2 response



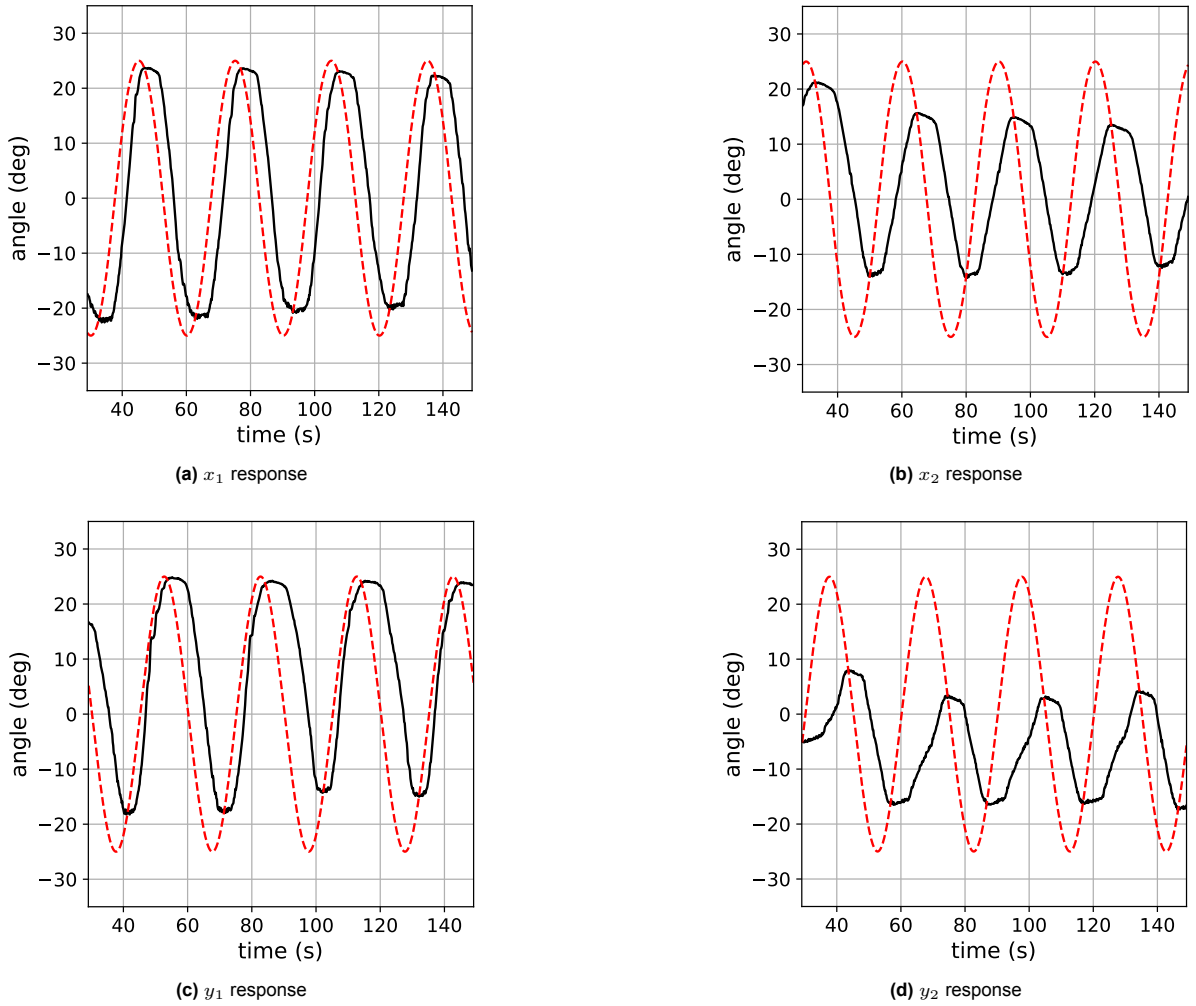
(c) y_1 response



(d) y_2 response

Figure H.2: Response of all axes to a sinusoidal reference input at $\frac{1}{60}$ Hz. The reference signal is shown in red, while the black curve denotes the filtered Hall sensor measurement.

H.3. Response at $\frac{1}{30}$ Hz



(a) x_1 response

(b) x_2 response

(c) y_1 response

(d) y_2 response

Figure H.3: Response of all axes to a sinusoidal reference input at $\frac{1}{30}$ Hz. The reference signal is shown in red, while the black curve denotes the filtered Hall sensor measurement.

TU Delft

Design and Control of a Soft-Robotic Tentacle Using Shape Memory Alloys

Ruben Tolenaars
MSc Student, TU Delft

June 1, 2026

Abstract

This work presents the design and control of a soft-robotic tentacle actuated by shape memory alloy (SMA) springs, inspired by the muscular arrangement of an octopus arm. A desktop-scale prototype consisting of three antagonistically actuated sections was developed, providing six rotational degrees of freedom. To achieve proportional control of the SMA actuators, a baseline PID controller was compared with a model-based feedforward–feedback (FF–FB) controller incorporating an inverse Preisach hysteresis model. The Preisach model was identified and experimentally validated on a dedicated single-actuator test bench. Results show that the model accurately captures major hysteresis loops but exhibits significant errors during minor loops and cannot account for rate-dependent hysteresis behavior. Consequently, the FF–FB controller underperformed relative to the PID controller, which demonstrated robust and reliable tracking on both the test bench and the complete tentacle prototype. These findings confirm that proportional closed-loop control of an SMA-actuated multi-section tentacle is feasible, while highlighting that high control accuracy remains challenging due to the non-linear, hysteretic nature of SMA actuators. Overall, SMAs offer a compact, silent, and mechanically simple actuation solution, with bandwidth and control accuracy identified as the primary limitations for future development.

Keywords— artificial muscles, biomimetic robotics, shape memory alloy, soft actuators

1. Introduction

Traditional robotic systems are typically constructed from rigid links and joints driven by electric motors to generate motion. Although such robots are precise, they are often mechanically complex due to the large number of required components and have only a limited number of degrees of freedom. In contrast, soft-robotics is an emerging field that focuses on the use of flexible materials—such as elastomers or alloys—for actuation. The compliant structure of soft robots enables a high, potentially infinite, number of degrees of freedom while maintaining relative mechanical simplicity.

A subtype of soft actuators—also known as artificial muscles—is shape memory alloys (SMAs). SMAs recover to a predefined shape upon heating above a certain transformation temperature, which is know as the *shape-memory-effect* [1–6]. This behavior comes from a reversible solid-state phase transformation between the martensite and austenite crystal structures. When an SMA wire or spring is mechanically deformed in its low-temperature martensitic phase and subsequently heated, it contracts toward its original shape, thereby generating mechanical work. As a result, SMAs function as thermally driven actuators, with nickel–titanium alloys—also known as nitinol—being the most extensively studied and widely available. The main advantage of using SMAs in robotic applications instead of conventional actuators, such as electric motors, is that SMAs can provide a mechanically simple actuation mechanism. Whereas electric motors consist of multiple components, an SMA-driven robot can achieve motion using only a single part. Furthermore, SMA operation is completely silent, free of vibrations, relatively low cost, and capable of high force output.

An interesting source of inspiration for robotic systems is the octopus. An octopus arm consists of three main muscle groups composed of striated muscle cells: transverse, longitudinal, and oblique muscles [7, 8], as shown in Figure 1. The oblique muscle fibers are arranged in clockwise (CW) and counterclockwise (CCW) layers. Contraction of the longitudinal muscles shortens the arm, while contraction of the transverse muscles elongates it, resulting in antagonistic behavior between the two groups. Activation of the CW and CCW oblique layers causes twisting in the opposite directions. Simultaneous contraction of the transverse and longitudinal muscles stiffens the arm, whereas non-uniform contraction of the longitudinal muscles causes the arm to bend toward the region with the lowest stiffness. By arranging

SMA actuators similarly to the musculature of a real octopus tentacle, a highly dexterous robot with mechanical simplicity and low cost can be designed.

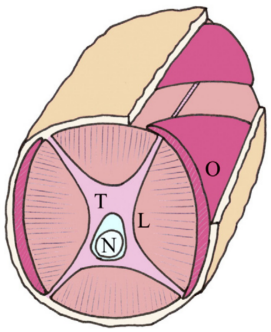


Figure 1: Cross-sections of an octopus arm showing muscular arrangement [8]. T: transverse muscle, L: longitudinal muscle, O: oblique muscle, N: central nerve cord.

A potential application of such a biomimetic tentacle is underwater inspection in hard-to-reach areas. Its compliant and highly dexterous nature allows it to wrap around irregular objects while reducing the risk of damaging sensitive marine life such as corals. This application is particularly attractive since the surrounding seawater could cool the SMAs more rapidly, thereby increasing their bandwidth. Another application is the handling of fragile products such as fruits or vegetables. Unlike rigid robots, a compliant tentacle can gently wrap around objects, reducing bruising or scratching. Such systems could also be used in search-and-rescue operations, where the tentacle navigates through debris while carrying microphones or cameras. Furthermore, the silent operation of SMAs may be advantageous in human-assistance applications where the continuous noise of electric motors is undesirable.

To design such a robotic tentacle, proportional control of the strain or displacement of the SMA actuators is required. However, achieving controlled behavior with SMAs is inherently challenging due to their non-linear thermomechanical response and hysteresis [1, 5, 9, 10]. Hysteresis is the phenomenon in which the state of the SMA depends not only on its current input, but also on its input history, thereby complicating proportional control. As a result, more advanced controllers—such as model-based controllers [5, 10]—are often implemented to achieve accurate strain control of SMA actuators. These controllers typically combine a feedback component, often implemented as a PID loop, with a feedforward component based on a model to compensate for non-linearity and hysteresis. Such hysteresis models—of which the empirical Preisach model is among the most widely used [11, 12]—are generally complex and require significant tuning effort, but have proven effective in advanced control strategies.

Numerous SMA-driven tentacle robots have been reported in the literature, but—to the author's knowledge—none contain multiple SMA actuators operating within a closed-loop control scheme. A notable example is the design presented in [3]. The tentacle is fully flexible, contains no rigid components, and closely mimics an octopus arm, with muscle orientations matching those of the natural counterpart. However, the SMAs are actuated using a full-on/full-off strategy rather than being driven to intermediate states. Similarly, the SMA-actuated tentacle or snake-arm robot described in [1] employs binary actuation. Some reported designs achieve strain control of a limited number of SMAs representing a section of a robotic tentacle. An example is described in [13] (see Figure 2), where three SMA springs are arranged antagonistically to control the section curvature. Although this is a promising concept, it does not demonstrate the feasibility of using SMAs in a complete multi-section tentacle design, as the reported system has limited load-bearing capability.

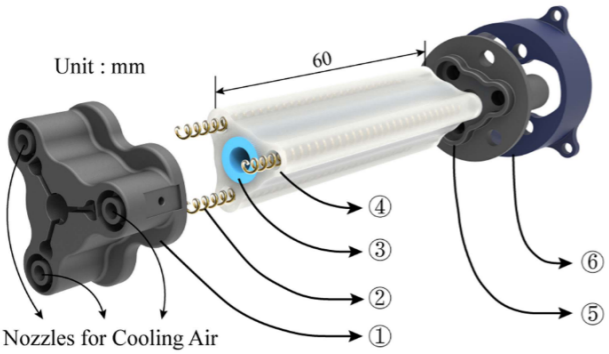


Figure 2: Soft tentacle section actuated by three SMA springs. Numbered annotations: (1) rigid top cap; (2) SMA springs (×3); (3) inner soft body; (4) outer soft body; (5) rigid bottom cap; (6) connection base. Adapted from [13].

Therefore, this work presents the design of a complete multi-section robotic tentacle in which each section can be proportionally controlled, and evaluates the feasibility of using SMAs as alternative actuators. Furthermore, the performance of a baseline PID controller is compared with that of a more advanced feedforward–feedback (FF–FB) control strategy containing a Preisach hysteresis model. The main contributions of this work are:

- Development of a desktop-scale SMA-actuated soft robotic tentacle with antagonistic actuation.
- Experimental validation of proportional control in a multi-section soft robotic tentacle.
- Comparison of PID and FF-FB control strategies for proportional SMA control.
- Experimental identification of a Preisach hysteresis model for SMA springs.

This paper is organized as follows. Section 2 provides background on SMA behavior, hysteresis modeling, and control methods. Section 3 describes the experimental setup and prototype tentacle design. Section 4 presents the implemented modeling and control strategies. Section 5 discusses the experimental results, followed by a discussion in Section 6. Finally, Section 7 concludes the paper.

2. Background

SMAs are a class of smart materials capable of returning to their original shape after deformation and have two unique properties: the shape memory effect (SME) and the superelastic effect (SE), also referred to as the pseudoelastic effect [4]. The SME is the ability to "memorize" a predefined shape, to which the material returns upon heating. The SE is the ability to undergo large deformations (5–10%) and fully recover from them without exhibiting plastic deformation [2–6]. These unique properties enable SMAs to function as actuators and are often formed into springs to increase stroke length at the expense of force output. Heating is often achieved by passing an electrical current through the material, known as Joule heating. At low temperatures and in an unloaded state, the material is in the *twinned martensite* phase. Under applied stress, it deforms into *detwinned martensite*, retaining this deformation after unloading. When heated above the austenite start temperature (A_s), a solid-state phase transformation to *austenite* begins, causing the actuator to recover its original shape. Full recovery occurs at the austenite finish temperature (A_f). Upon cooling from the martensite start (M_s) to the martensite finish (M_f) temperature, the material transforms back to martensite. The resulting major hysteresis loops—illustrated in Figure 3—exhibit non-linear, hysteretic behavior and depends on the applied actuator stress. Thermal cycling to an intermediate temperature—where the material consists of a mixed phase of *martensite* and *austenite*—results in the formation of a minor hysteresis loop. The combination of non-linearity and stress-dependent hysteresis makes accurate strain control challenging.

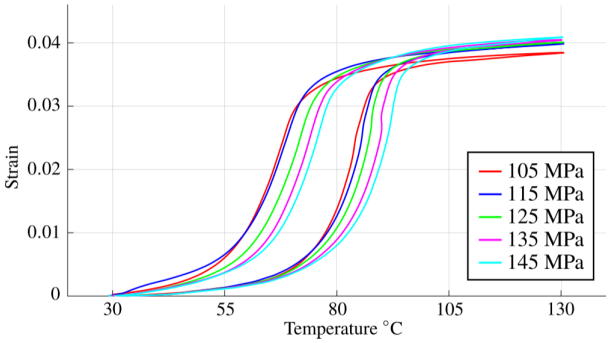


Figure 3: Strain–temperature major hysteresis loops of a 127 μm Nitinol SMA wire at five different stress levels. Reproduced from [5].

Arguably the most popular control strategy reported in the literature combines a PID controller as the feedback component with an empirical model as the feedforward component [14]. Intuitively, the feedforward component compensates for hysteresis and drives the actuator toward the desired strain, while the feedback component compensates for modeling errors and adjusts small reference errors. Many such strategies implement an inverse Preisach model that outputs the actuator temperature required to achieve a desired reference strain for a given actuator load [15, 16]. Other approaches use the Prandtl–Ishlinskii model to generate this temperature signal [17]. The main drawback of these strategies is that the models require calibration, which is often delicate and time-consuming. Moreover, the models frequently require multiple state estimates for online operation—such as actuator temperature and load—thereby increasing system complexity due to the need for additional sensors. As a result, many existing control strategies implement a pure PID controller without a feedforward component and demonstrate consistent SMA strain control performance. However, since PID is a linear controller, it may be inadequate for controlling the non-linear behavior of SMAs [10,18,19]. To compare the performance of a pure PID controller with a more advanced feedforward–feedback (FF–FB) controller, both strategies are implemented and analyzed in this work. The Preisach hysteresis model is selected for the FF–FB control scheme due to its widespread use in SMA applications.

The classical Preisach model of hysteresis [11] is used to describe single-input single-output (SISO) systems. The fundamental building blocks of the Preisach model are hysteresis operators called *hysterons* [5, 10–12], where the output of a hysteron $\gamma_{\alpha\beta}$ is defined as

$$\gamma_{\alpha,\beta}[T] = \begin{cases} 1, & \text{if } T(t) > \alpha, \\ 0, & \text{if } T(t) < \beta, \\ \text{previous state}, & \text{if } \beta \leq T(t) \leq \alpha, \end{cases} \tag{1}$$

with $T(t)$ the temperature input, and α and β the upper and lower thresholds, respectively. The Preisach model computes the actuator strain by summing the output of all hysterons on the triangular plane $\{(\alpha, \beta) \mid \beta_0 \leq \beta \leq \alpha \leq \alpha_0\}$, where α_0 and β_0 are the austenite and martensite finish temperatures, respectively, and multiplying each by a weight function $\mu(\alpha, \beta)$ [5]. Therefore, the Preisach strain is defined as

$$\epsilon(t) = \iint_{\alpha \geq \beta} \mu(\alpha, \beta)\, \gamma_{\alpha,\beta}[T](t)\, \mathrm{d}\alpha\, \mathrm{d}\beta. \tag{2}$$

The weight function must be identified empirically, and numerous methods exist. One example is the approach described in [20], where multiple experiments are performed and the weights are identified using a least-squares optimization algorithm. In this work, a *first-order-reversal-curve* (FORC) method described in [5] is used, as it requires a minimal number of experiments and therefore reduces calibration time. Since the Preisach model is inherently SISO, while SMA strain depends on both temperature and the applied time-varying stress, the model must be extended to a multi-input single-output (MISO) system. Therefore, the Preisach model in this work is expanded by calibrating five weight functions at equally spaced stress levels. During online operation, the model tracks the state of each hysteron and computes the strain using bilinear interpolation between the identified stress levels.

3. System Design and Experimental Setup

The performance of the PID and FF–FB controllers was initially analyzed using a single SMA actuator rather than a complete tentacle design. Therefore, the experimental setup of this work consists of two distinct systems: a test bench containing a single SMA actuator and a prototype tentacle containing multiple antagonistic actuators.

3.1. Single SMA Test Bench

The purpose of the test bench is to identify the Preisach model for the FF–FB controller and analyze the strain-tracking performance of both controllers. In the setup—shown in Figure 4—a nitinol spring is connected to a rope running over a spool (pink). The spool is coupled to a capacitive encoder with 8192 counts per revolution [21] to measure the strain, defined as

$$\epsilon = \frac{\text{max extension} - u(x)}{L_0}, \tag{3}$$

where $u(x)$ is obtained from the encoder measurements, accounting for the spool diameter. Here, L_0 is the unloaded actuator length in saturated austenite, and the maximum extension corresponds to the maximum allowable extension in saturated martensite without permanently damaging the actuator. Furthermore, the test bench includes a thermocouple [22], a PWM module for Joule heating [23], and a microcontroller [24] for reading sensors and PWM actuation. The microcontroller is connected to a desktop PC, which runs the model and both controllers in Python [25]. This setup enables identification of the Preisach model, evaluation of the PID and FF–FB controller performance, and selection of a suitable controller for the complete tentacle prototype.

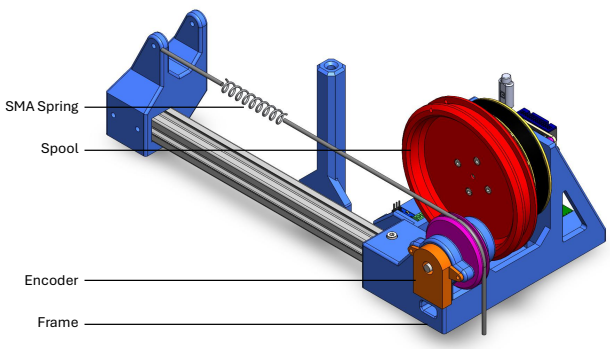


Figure 4: CAD representation of the SMA test bench.

3.2. Tentacle Prototype

The proposed tentacle (shown in Figure 5) consists of three 3D-printed sections, each providing two rotational degrees of freedom (x_i, y_i for $i = 1, \ldots, 3$) through a Hooke's joint [26]. Each section is driven by four SMA actuators arranged in antagonistic pairs. The SMAs used in the design are nitinol springs selected for their high force output and large recoverable strain. The joint angle of each axis is limited to $-30°$ to $30°$, and the structure is designed such that these limits correspond to the maximum SMA extension (saturated martensite) and minimum SMA length (saturated martensite), respectively.

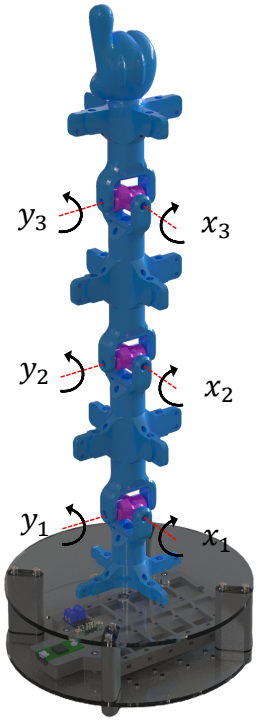


Figure 5: CAD representation of the designed tentacle. The axes x_i and y_i for $i = 1, \ldots, 3$ denote the angles of sections 1–3.

Although three actuators per section would be sufficient in principle, four actuators are used to decouple the two rotational axes: each axis is driven by an independent antagonistic pair, ensuring that actuation of one axis does not directly affect the load on the other. This simplifies controller analysis and tuning and results in a compliant structure with six total degrees of freedom across the full tentacle.

Joint angles are measured using low-cost analog Hall effect sensors [27], where a magnet embedded in the rotating joint varies its distance to the sensor as the section deflects. This provides a compact and lightweight sensing solution at the cost of reduced accuracy and increased noise, which is acceptable given the focus on control feasibility rather than high-precision metrology. The actuation and control architecture is similar to the test bench setup: a microcontroller [24] reads and filters sensor signals and transmits them to Python at 100 Hz. The computed control signals are

returned to the microcontroller as duty cycle commands at 40 Hz, which drive PWM modules [23] powering the SMA actuators.

4. SMA Modeling and Control Methods

4.1. SMA Modeling

The weight function of the Preisach model was identified using the FORC method described in [5], which requires only a minimal number of experiments and therefore reduces calibration time. A FORC increment is obtained by monotonically heating the actuator from its martensite saturation temperature β_0 to α' and then monotonically cooling to β'. Intuitively, this represents the strain "released" by the actuator when cooling from α' to β', assuming the measurement started from fully saturated martensite. This can be represented geometrically using the Preisach plane, as shown in Figure 6a and Figure 6b. A FORC increment corresponds to the region in the Preisach plane—defined as the triangle enclosed by the lines $\alpha = \alpha'$, $\beta = \beta'$, and $\alpha = \beta$—denoted by $\Delta(\alpha', \beta')$. Thus, $\Delta(\alpha', \beta')$ represents the portion of the Preisach plane that is wiped out when the temperature decreases from $\alpha = \alpha'$ to $\beta = \beta'$, and it is equivalent to the surface integral of the weight function over the region in which the hysterons switch off after this temperature reversal.

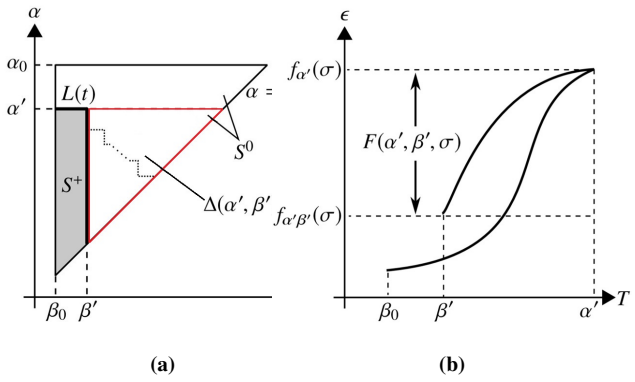


Figure 6: Geometrical interpretation of a FORC strain increment. **(a)** α_0 and β_0 denote the austenite and martensite saturation temperatures, respectively. The red triangle $\Delta(\alpha', \beta')$ represents the region that is wiped out from the Preisach plane when the temperature is increased from β_0 to α' and subsequently decreased from α' to β'. **(b)** Strain–temperature relationship of a FORC increment with reversal-point temperature α' and end-point temperature β'. The strain increment between $f_{\alpha'}$ and $f_{\alpha',\beta'}$, denoted as $F(\alpha', \beta')$, corresponds to the surface integral of the weight function over the region where the hysterons switch off after the first temperature reversal, that is, the triangle $\Delta(\alpha', \beta')$. Adapted from [5].

Therefore, a FORC increment at a constant stress level σ is defined as

$$F(\alpha', \beta', \sigma) = f_{\alpha'}(\sigma) - f_{\alpha',\beta'}(\sigma), \tag{4}$$

where $f_{\alpha'}(\sigma)$ denotes the strain at temperature α', and $f_{\alpha',\beta'}(\sigma)$ denotes the strain after cooling from α' to β'. Using this definition, the Preisach strain from Equation 2 can be expressed as a finite sum of measurable FORC increments:

$$\epsilon(t,\sigma) = \begin{cases} \displaystyle\sum_{k=1}^{n(t)-1} \Big[F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\Big] + F(\alpha_n, \beta_{n-1}) \\ \qquad\qquad \text{if } \dot{T}(t) > 0 \quad \text{(heating)} \\ \displaystyle\sum_{k=1}^{n(t)} \Big[F(\alpha_k, \beta_{k-1}) - F(\alpha_k, \beta_k)\Big] \\ \qquad\qquad \text{if } \dot{T}(t) < 0 \quad \text{(cooling)} \end{cases} \tag{5}$$

The FORC measurements were obtained by discretizing the Preisach plane over the triangular domain $\{(\alpha, \beta) \mid 25°\text{C} \leq \beta \leq \alpha \leq 64°\text{C}\}$ using a uniform step size of 3°C along both axes. For each reversal temperature α' on the discretized α-axis, the actuator was first heated from β_0 to α' and subsequently cooled monotonically back to β_0. During cooling, FORC increments $F(\alpha', \beta', \sigma)$ were identified for every discretized temperature $\beta' < \alpha'$, resulting in 105 FORC measurements for a constant stress level.

4.2. Control Methods

The PID controller was implemented on both the test bench and the prototype tentacle. On the test bench, the controller directly outputs a PWM duty cycle based on the error between the reference strain and the measured encoder strain, resulting in a straightforward SISO implementation.

For the antagonistic tentacle configuration, the PID controller must be extended since each axis is driven by two actuators. Based on a method by [28], the PID output (see Figure 8) is added to a constant feedforward duty cycle K_{ff} for SMA A and subtracted from K_{ff} for SMA B within each antagonistic pair. The term K_{ff} defines the baseline heating level and thereby the effective stiffness of the system, as higher values correspond to increased actuator temperatures. Without this feedforward term, no heating occurs near zero error, resulting in low stiffness and reduced system stability. All controller parameters, including K_{ff} for the tentacle setup, are tuned empirically. For the tentacle, the controller gains were tuned to prioritize fast response at the expense of overshoot.

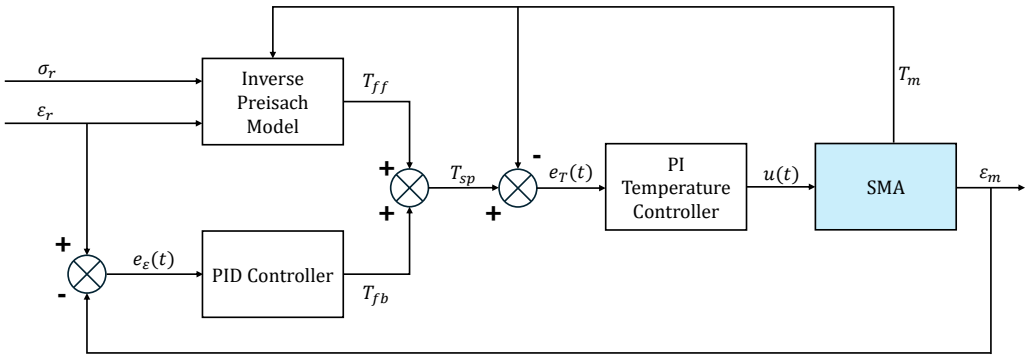


Figure 7: Control scheme of the FF–FB controller implemented in the test bench. The reference strain is denoted by ϵ_r, the reference stress by σ_r, and the strain error by $e(t)$. The inverse model generates the feedforward temperature T_{ff} corresponding to the desired reference stress and is updated only upon changes in the reference. The PID controller computes the feedback temperature T_{fb} based on $e(t)$ and is activated once the temperature tracking error is within $\pm 1^\circ$C of T_{ff}. The temperature T_m measured by the thermocouple is subtracted from the reference temperature T_{sp} and fed as the error signal $e_T(t)$ to the PI controller, which outputs the control signal $u(t)$ as a PWM duty cycle in the range $[0, 1]$. The measured temperature is fed into the inverse Preisach model, while the measured strain ϵ_m (from the encoder) is used to compute $e_\epsilon(t)$.

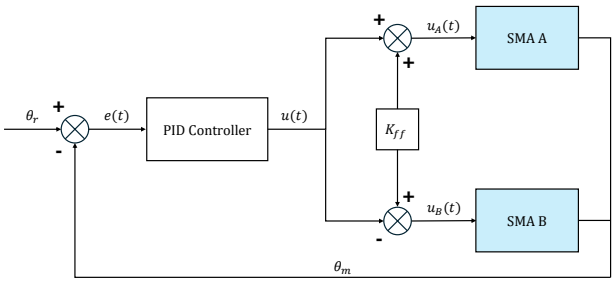


Figure 8: Control scheme of the prototype tentacle. The reference angle θ_r is compared with the measured angle θ_m to compute the error $e(t)$, which is used by the PID controller to generate $u(t)$. This signal is combined with the feedforward term K_{ff} to produce the actuator inputs $u_A(t)$ and $u_B(t)$ for the antagonistic SMA pair.

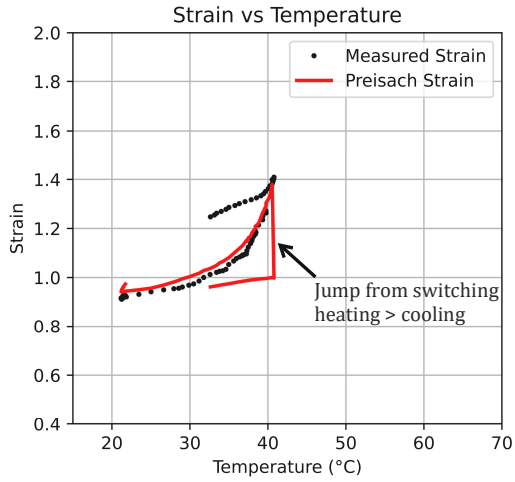
The FF–FB controller was implemented only in the test bench, primarily due to its limited performance (as will become clear in the next chapter) and its requirement for additional state estimates. The identified Preisach model was inverted using a bisection search method [29] to compute the temperature required to achieve a given reference strain. The resulting feedforward term from the inverse model and the feedback term from a PID controller are combined and applied to a PI temperature controller, forming a cascaded control structure illustrated in Figure 7. The feedforward temperature T_{ff} from the inverse model and the feedback temperature T_{fb} from the PID controller are summed to obtain the setpoint temperature T_{sp}. The measured thermocouple temperature T_m is subtracted from this setpoint to define the temperature error $e_T(t)$, which is used by the temperature controller to track T_{sp} by generating a PWM duty cycle $u(t)$. In addition, the inverse Preisach model uses T_m, the reference stress σ_r, and the reference strain ϵ_r to compute T_{ff}. Intuitively, the feedforward term drives the system towards the desired reference, while the feedback term compensates for model inaccuracies and external disturbances. The measured strain ϵ_m, obtained from the encoder, is used to compute the strain error $e_\epsilon(t)$, which is processed by the PID controller to generate the feedback temperature term. Note that the PID controller in this architecture differs from the one in the pure PID setup: here it outputs a temperature reference, whereas in the previous implementation it directly outputs a PWM duty cycle.

The inverse Preisach model updates the feedforward term only when ϵ_r changes. Furthermore, to prevent undesired integrator wind-up, the PID controller is only activated when the setpoint temperature is within $\pm 1^\circ$C of the feedforward temperature. Outside this range, the PID controller remains idle. With this control strategy, the feedforward term effectively replaces part of the integral action of the PID controller. Instead of relying only on feedback—where the integral term must converge to the steady-state value within the hysteresis loop—the required control effort is partially predicted a priori by the feedforward component. Therefore, this strategy theoretically allows the controller to achieve better tracking performance than a pure PID controller.

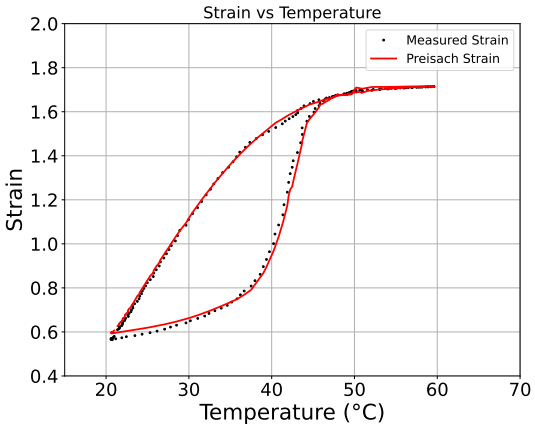
5. Experimental Results

5.1. Preisach Model Evaluation

Analysis of the identified Preisach model revealed that the FORC identification method could not accurately capture the heating trajectories of the strain–temperature hysteresis



(a) Minor hysteresis loop of the combined model for a load of 2.5 N. The actuator is heated from approximately 20 °C to 40 °C and subsequently cooled to approximately 32 °C.



(b) Major hysteresis loop of the combined model for a load of 3.0 N. The actuator is heated from martensite saturation to austenite saturation and subsequently cooled back to martensite saturation.

Figure 9: Strain–temperature relationships of the model (red) using both the heating and cooling FORC data, compared with experimental data (black).

loops. This was due to the fact that the FORC increments were measured during cooling and therefore did not contain sufficient information about the thermomechanical heating behavior of the SMA. To improve model performance, the FORC identification method was extended to identify two distinct Preisach models for each stress level: one based on cooling FORC data and one based on heating FORC data.

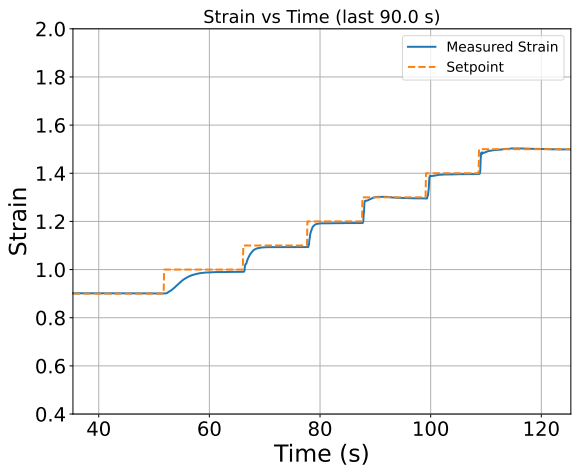
An example of the resulting model behavior is shown in Figure 9a, where the model is heated from the martensite saturation temperature and subsequently cooled. At the temperature reversal, i.e., when switching from the heating model to the cooling model, the predicted strain exhibits a sudden drop to a low value. This behavior originates from the inability of the cooling model to accurately capture the heating response of the SMA. During heating from the martensite saturation temperature to approximately 40°C, the cooling model predicts a strain that is too low. When the temperature is then reversed, this incorrect strain value is used as the reversal reference, which leads to a poor prediction of the subsequent minor hysteresis loop.

The resulting model shows excellent agreement for major hysteresis loops (see Figure 9b) but poor performance for minor hysteresis loops. Since the strain prediction is based on separate heating and cooling FORC datasets, the model cannot accurately reproduce minor loops, which require a weight function that is consistent for both heating and cooling. This suggests that the FORC identification method is not well suited for identifying the Preisach weight function. Furthermore, it was found that the SMA hysteresis loops varied with the heat input rate. Faster actuation produced steeper hysteresis loops, and the identified model could not capture this rate-dependent hysteresis behavior.

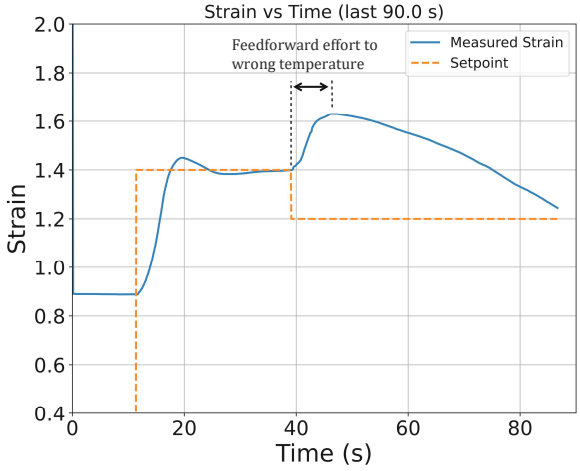
5.2. Single Actuator Control Results

The performance of the PID controller was analyzed using the test bench by tracking step inputs and sinusoidal inputs. Overall, the tracking performance of the PID controller shows surprisingly accurate results. Upon reaching the setpoint, the output stabilizes without showing chattering. Furthermore, the controller gains were easy to tune, and no significant oscillations are observed upon reaching the setpoint. As stated in the literature, a limitation of the PID controller is that it cannot capture the non-linear and hysteretic behavior of the actuator. This can be confirmed by the incremental step-up response shown in Figure 10a. For the first step input–where the strain setpoint is increased from 0.9 to 1.0–the response of the system is significantly slower than for the last step input–where the strain setpoint is increased from 1.4 to 1.5. This is a result of the highly non-linear strain–temperature behavior of SMA actuators. During heating, the actuator used in this work exhibits high sensitivity in the strain range between 1.2 and 1.6, meaning that a small increase in the PID output $u(t)$ results in a large increase in strain. In contrast, at lower strain ranges, a relatively large control input is required to obtain a similar strain response. This difference in sensitivity explains the variation in settling time observed in Figure 10a.

The overall performance of the FF–FB controller was worse than that of the PID controller, mainly due to large model errors during minor hysteresis loops. In Figure 10b, the actuator is heated from (cold) saturated martensite to an intermediate temperature, resulting in a strain of 1.4. During this trajectory, an overshoot occurs at approximately $t = 20$ s, after which the controller cools the actuator back to the setpoint. This reversal introduces a minor hysteresis loop in



(a) Response of the PID controller.



(b) Response of the FF–FB controller.

Figure 10: Time–strain step responses of the actuator under a fixed load of 2.5 N. The orange curve denotes the strain setpoint, and the blue curve the measured strain.

the model. As a result, for the setpoint of 1.2, the model outputs a feedforward temperature that is too high, causing the controller to initially increase the strain instead of decreasing it. Therefore, non-monotonic heating or cooling can introduce large model errors and the controllers shows reliable tracking performance only for major hysteresis loops. Furthermore, the FF–FB controller showed slower settling times due to its cascaded control structure and the limited response speed of the temperature controller. This slow response was primarily caused by the low bandwidth (below 10 Hz) of the thermocouple and could be improved by using higher-grade temperature sensors. A comparison of the evaluated controllers is provided in Table 1.

Criterion	**PID**	**FF–FB**
Reference tracking performance	*Good*	*Poor*
Settling time	*Good*	*Medium*
Steady-state error	*Good*	*Good*
Implementation complexity	*Good*	*Poor*
Computational requirements	*Good*	*Medium*
Number of required state estimates	*Good*	*Poor*
Tuning effort	*Good*	*Poor*
Flexibility and adaptability	*Good*	*Medium*
Response consistency	*Poor*	*Good*

Table 1: Qualitative comparison of the evaluated controllers based on performance and implementation criteria.

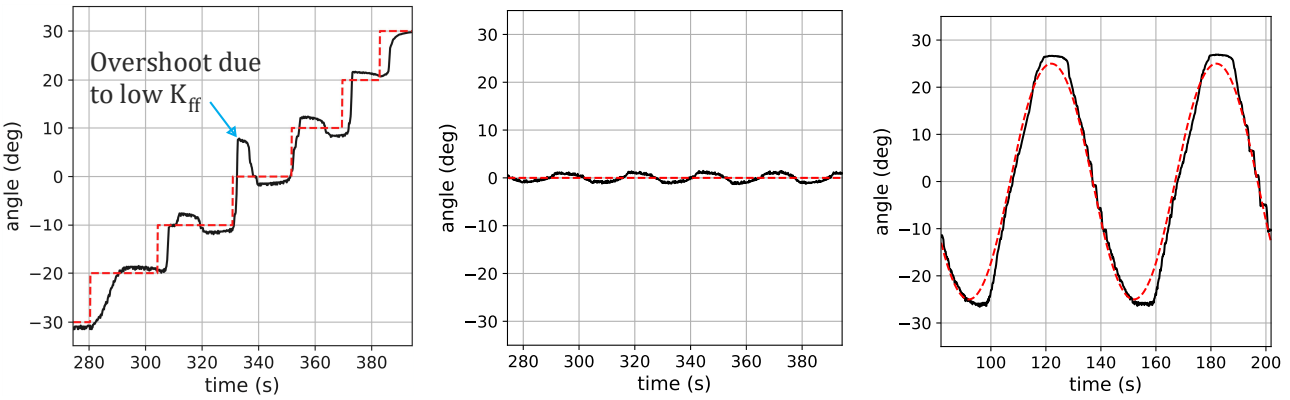
Based on these results, the PID controller was selected for implementation in the complete tentacle prototype.

5.3. Tentacle Results

Due to the superior performance of the PID controller and its limited number of required state estimates and therefore sensors, only the PID controller was implemented in the prototype tentacle. Overall, the PID controller was easy to tune and showed reliable and robust tracking performance, although some overshoot is present in the response during incremental step references (see Figure 11a). When this reference reaches $0°$, the torque direction of axis x_1 reverses, and the load suddenly shifts to the colder, less stiff left actuator. Since this actuator requires time to heat up and generate stiffness, an overshoot occurs. This effect is most pronounced for x_1, as it experiences the largest torque change due to the mass and inertia of the tentacle and increasing K_{ff} decreases the overshoot. However, to prevent overheating of the PWM modules, K_{ff} was kept low intentionally. The effect of the low K_{ff} value–and therefore low system stiffness–becomes especially clear for constant references, as shown in Figure 11b, where oscillations are present around the setpoint. It was observed that increasing K_{ff} reduces these oscillations, likely due to increased stiffness at higher actuator temperatures, which is consistent with the method described in [28].

Another important observation is that, for each axis, $0°$ represents an unstable equilibrium, meaning that even small disturbances cause the system to move further away from this position. This behavior is caused by gravity as well as the load introduced by the actuators of the orthogonal axis. For example, when actuating axis y_1 while keeping x_1 unactuated (cold), axis x_1 will rotate toward either $-30°$ or $30°$, depending on its initial condition. This indicates that the axes are not fully independent and helps explain the oscillations observed in Figure 11b.

The response to a sinusoidal reference of $\frac{1}{60}$ Hz is shown in



(a) x_1 incremental step response.

(b) y_1 response to a constant reference.

(c) x_1 response to a sinusoidal reference of $\frac{1}{60}$ Hz.

Figure 11: Responses of the tentacle axes to reference inputs. The reference signal is shown in red, while the black curve denotes the filtered Hall sensor measurement.

Figure 11c, and shows satisfactory tracking performance. In general, the controller tracks slowly varying references well, but performance degrades at higher frequencies due to the limited bandwidth of the SMA actuators. Therefore the performance of the tentacle was limited mainly by the slow cooling times of the SMAs and limitations of the PWM modules, rather than the controller itself.

6. Discussion

The PID controller outperformed the FF–FB controller in the evaluated experiments for several reasons. First, the model used in the FF–FB control structure contained significant errors, resulting in poor feedforward predictions. These errors originated from limitations of the FORC identification method, which could not fully capture the SMA behavior, as well as from the actuator's sensitivity to heating rate, which the Preisach model cannot capture. Potential improvements include the use of a model that accounts for rate-dependent hysteresis or an alternative identification approach. Another possibility is to use the model's derivative trend—that is, the slope of the strain–temperature relationship—for gain scheduling of the PID controller. This would keep the simplicity of the PID structure while compensating for its inconsistent response. The results (e.g., Figure 9a) showed that although the model often produced large strain prediction errors, the derivative of the strain–temperature relationship remained similar to the measured data. Therefore, the model may still provide useful information within a simple control framework despite its poor accuracy during minor hysteresis loops. Second, the cascaded structure of the FF–FB controller increased tuning complexity and resulted in slower response. Third, the FF–FB controller required a large number of state estimates, thereby increasing overall system complexity.

In contrast, the PID controller provides a simple, robust, and reliable method for controlling SMA actuators in a robotic tentacle without requiring numerous sensors. Its main limitation is the lack of system knowledge, which leads to larger tracking errors due to the compliant nature of SMAs and inconsistent response caused by the non-linear strain–temperature relationship. For robotic tentacle applications, the results indicate that proportional SMA control is feasible, but achieving high-accuracy control remains challenging. Consequently, the suitability of SMAs depends strongly on the intended application and its performance requirements.

Improving controller accuracy will likely require more advanced control strategies that depend on additional state estimates, such as actuator temperature, strain, and stress. However, one of the primary advantages of SMAs in robotics is their mechanical simplicity. Obtaining these state estimates typically requires additional sensors, increasing system complexity and partially negating this advantage. Although some states can be estimated without dedicated sensors, implementation remains challenging. For example, actuator temperature can be estimated using a heat-transfer model, while strain can be inferred from actuator resistance using the existing actuation wiring. However, such sensorless estimation methods require large implementation effort. In particular, resistance-based strain estimation requires identification of the strain–resistance relationship, which is hysteretic [30]. Therefore, improving the tracking accuracy of the prototype tentacle remains a challenging task, suggesting that SMAs are best suited to applications where mechanical simplicity is more important than high control accuracy.

Instead of implementing a complex control strategy, overall system performance could be improved by increasing actuator bandwidth. The prototype exhibited relatively slow response, primarily due to the slow cooling rates of the SMAs. This can be improved by replacing natural convection with forced convection. Alternatively, conductive cooling methods, such as circulating cooling oil or operation in an un-

derwater environment, could be employed. Another approach is to use multiple smaller SMAs in parallel, thereby increasing the surface-area-to-volume ratio and improving heat transfer. Many such methods have been reported in the literature [3, 31] and would arguably provide a more straightforward route to improved system performance than increasing controller complexity.

Overall, SMAs offer a simple, compact, silent, and compliant alternative to conventional actuators, with their primary limitations being low bandwidth and limited control accuracy. For applications where these limitations are acceptable, SMAs may provide a simple and cost-effective solution. It should be noted that this work did not investigate actuator fatigue or degradation of the shape memory effect under repeated use. However, the literature suggests that overheating or overstretching SMAs can degrade performance and should therefore be considered in future work [1]. Preventing such degradation may require monitoring actuator states, such as temperature, which would likely require additional sensors and increase system complexity.

7. Conclusion

This work presented the design and control of an SMA-actuated soft robotic tentacle inspired by the muscular arrangement of an octopus arm. A test bench was developed to identify a Preisach hysteresis model and evaluate proportional strain control of a single SMA actuator. In addition, a multi-section tentacle prototype containing antagonistically arranged SMA springs was designed and used to investigate the feasibility of proportional closed-loop control in a complete robotic system. The performance of a baseline PID controller was compared with that of a model-based feedforward–feedback (FF–FB) controller incorporating an inverse Preisach model.

The results showed that the identified Preisach model accurately reproduced major hysteresis loops but exhibited significant errors during minor loops and could not capture the rate-dependent behavior of the SMA actuator. Consequently, the FF–FB controller provided limited tracking performance despite its increased complexity and additional state-estimation requirements. In contrast, the PID controller demonstrated robust and reliable behavior on both the (single-actuator) test bench and the prototype tentacle, while requiring substantially less tuning effort and state estimates. These findings indicate that proportional control of SMA-actuated tentacles is feasible using relatively simple control strategies, although achieving high control accuracy remains challenging due to hysteresis, non-linearity, and the compliant nature of SMAs.

Overall, SMAs provide a compact, low-cost, compliant, and mechanically simple actuation solution for soft robotic tentacles. However, their practical applicability depends strongly on the requirements of the intended application. Future work should focus on improving actuator bandwidth and investigating control strategies capable of accounting for rate-dependent hysteresis while preserving the simplicity that makes SMA-based robotic systems attractive.

References

[1] J. Romančík, Ľ. Miková, P. Šarga, T. Kelemenová, and M. Kelemen, "Shape memory alloy actuators in robotics," *Actuators*, vol. 15, no. 3, p. 162, 2026.

[2] G. Rizzello and P. Motzki, "Smart materials for mini-actuators," in *Endorobotics: Design, R and D and Future Trends*, pp. 117–163, Elsevier, 2022.

[3] M. Cianchetti, A. Licofonte, M. Follador, F. Rogai, and C. Laschi, "Bioinspired soft actuation system using shape memory alloys," *Actuators*, vol. 3, no. 3, pp. 226–244, 2014.

[4] A. Brotzu, V. Di Cocco, F. Iacoviello, S. Natali, and C. Vendittozzi, *Latest attainments*, pp. 53–76. Elsevier, 2021.

[5] J. Z. Ge, L. Chang, and N. O. Pérez-Arancibia, "Preisach-model-based position control of a shape-memory alloy linear actuator in the presence of time-varying stress," *Mechatronics*, vol. 73, p. 102452, 2021.

[6] S. Ameduri, "Design of sma-based structural actuators," in *Shape memory alloy engineering: For aerospace, structural, and biomedical applications*, pp. 485–524, Elsevier, 2021.

[7] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 2008.

[8] M. Cianchetti, A. Arienti, M. Follador, B. Mazzolai, P. Dario, and C. Laschi, "Design concept and validation of a robotic arm inspired by the octopus," *Materials Science and Engineering: C*, vol. 31, no. 6, pp. 1230–1239, 2011.

[9] R. Tolenaars, "Literature review: Artificial muscles for a biomimetic octopus," tech. rep., Delft University of Technology, Delft, The Netherlands, 2025. MSc literature review.

[10] K. K. Ahn and N. B. Kha, "Modeling and control of shape memory alloy actuators using preisach model, genetic algorithm and fuzzy logic," *Mechatronics*, vol. 18, no. 3, pp. 141–152, 2008.

[11] I. D. Mayergoyz and G. Friedman, "Generalized preisach model of hysteresis," *IEEE Transactions on Magnetics*, vol. 24, no. 1, pp. 212–217, 1988.

[12] A. Ktena, D. I. Fotiadis, P. D. Spanos, and C. V. Massalas, "A preisach model identification procedure and simulation of hysteresis in ferromagnets and shape-memory alloys," *Physica B: Condensed Matter*, vol. 306, no. 1–4, pp. 84–90, 2001.

[13] X. An, Y. Cui, H. Sun, Q. Shao, and H. Zhao, "Active-cooling-in-the-loop controller design and implementation for an sma-driven soft robotic tentacle," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2325–2341, 2023.

[14] X. Li, B. Zhang, D. Zhang, X. Zhao, and J. Han, "Disturbance compensation-based output feedback adaptive control for shape memory alloy actuator system," *International Journal of Advanced Robotic Systems*, vol. 18, no. 1, 2021.

[15] S. Majima, K. Kodama, and T. Hasegawa, "Modeling of shape memory alloy actuator and tracking control system with the model," *IEEE Transactions on Control Systems Technology*, vol. 9, 2001. PID + feedforward (static hysteresis model).

[16] J.-H. Lin and M.-H. Chiang, "Tracking control of a magnetic shape memory actuator using an inverse preisach model with modified fuzzy sliding mode control," *Sensors*, vol. 16, no. 9, p. 1368, 2016.

[17] M. R. Zakerzadeh and H. Sayyaadi, "Precise position control of shape memory alloy actuator using inverse hysteresis model and model reference adaptive control system," *Mechatronics*, vol. 23, pp. 1150–1162, 2013. Fig 15: hysteresis model (Prandtl-Ishlinskii) as feedforward term and PI controller as feedback control scheme.

[18] E. Shi, X. Zhong, T. Wang, X. Li, C. Bu, and X. Zhao, "Adaptive control for shape memory alloy actuated systems with applications to human–robot interaction," *Frontiers in Neuroscience*, vol. 18, 2024.

[19] Villoslada, N. Escudero, F. Martín, A. Flores, C. Rivera, M. Collado, and L. Moreno, "Position control of a shape memory alloy actuator using a four-term bilinear pid controller," *Sensors and Actuators A: Physical*, vol. 236, pp. 257–272, 2015.

[20] S. A. Shabalovskaya, "Electronic structure and properties of niti shape memory alloy," *Physica B: Condensed Matter*, vol. 312-313, pp. 1003–1005, 2002.

[21] Same Sky, "Amt102-v datasheet: Amt10 series modular incremental encoder," tech. rep., Same Sky, 2024. Rev. 1.15, September 12, 2024.

[22] M5Stack, *Unit KMeter — K-type Thermocouple Sensor with I²C Interface*. M5Stack, 2026. Online documentation.

[23] Alpha & Omega Semiconductor, *AOD4184A: 40V N-Channel MOSFET Datasheet*. Alpha & Omega Semiconductor Inc., Sunnyvale, CA, USA, 9 2023. Rev. 1.0.

[24] PJRC, "Teensy development boards." `https://www.pjrc.com/teensy/`, 2026. Accessed February 2026.

[25] Python Software Foundation, “Python programming language,” 2026. Accessed: 2026-05-27.

[26] A. Mills, “Robert hooke’s ’universal joint’ and its application to sundials and the sundial-clock,” *Notes and Records of the Royal Society*, vol. 61, no. 2, pp. 219–236, 2007.

[27] Nanjing Ouzhuo Technology Co., Ltd., “OH49E Series Linear Hall-Effect IC,” datasheet, Nanjing Ouzhuo Technology Co., Ltd., n.d. Accessed: 2026-04-28.

[28] J. Ko, “Fuzzy PWM-PID control and shape memory alloy actuator design for cocontracting antagonistic muscle pairs in an artificial finger,” master of applied science, University of Victoria, Victoria, BC, Canada, May 2011. Department of Mechanical Engineering.

[29] J. R. Chasnov, *Numerical Methods*. Hong Kong: LibreTexts™, 2025. Compiled on 06/09/2025.

[30] B. Lynch, X.-X. Jiang, A. Ellery, and F. Nitzsche, “Characterization, modeling, and control of ni-ti shape memory alloy based on electrical resistance feedback,” *Journal of Intelligent Material Systems and Structures*, vol. 27, no. 18, pp. 2489–2507, 2016.

[31] Deft Dynamics, “Smart material actuators,” 2018. Accessed: 2026-05-19.