POLITECNICO DI TORINO

UNIVERSIDADE DE COIMBRA

Department of CONTROL AND COMPUTER ENGINEERING (DAUIN)

Master Degree in Mechatronic Engineering

Control methods for liquid-gas phase transition soft actuators: from simulation to implementation.

Supervisors:
PhD Diogo Pereira da Fonseca
Prof. Pedro Mariano Simões Neto
Prof. Massimo Canale

Candidate:
Alice Allegretti

31 October 2024

Contents

1 Introduction

Soft robotics is an emerging field of robotics that aims to develop robots made of soft and flexible compliant materials which can bend, stretch, and deform like natural organisms. The concept of soft robotics is inspired by the natural world, where soft-bodied organisms, such as invertebrates, octopus[74], caterpillars[38] and jellyfish[72] have the ability to adapt to various environments and perform complex tasks.[45]

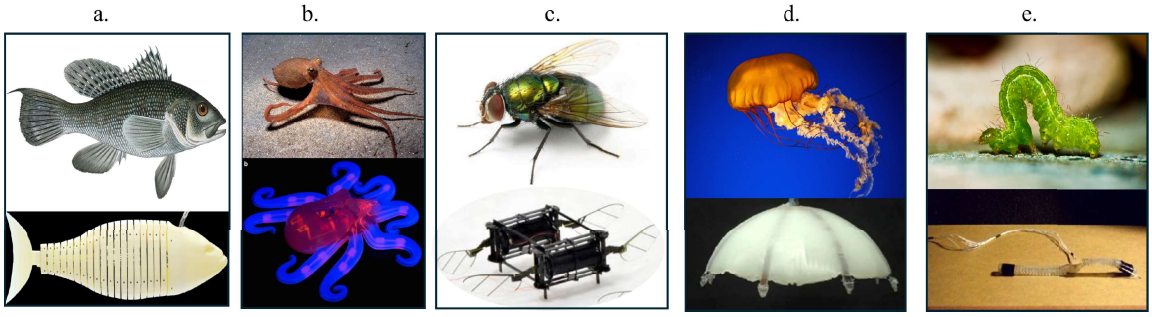


Figure 1: Images of animals and their soft robots inspired counterparts
(a) Bass fish & SMA actuated flexible fish robot[16], (b) Octopus & octobot [74]
(c) Flying insect & insect-scale aerial robot [11], (d) Jellyfish & Robojelly [72], (e) Caterpillar & GoQBot [38]

1.1 Motivations

The motivations for pursuing the research of soft robotics are related to the greater advantages they have with respect to their rigid counterparts. The main advantages of soft robotics in general are:

1. **Degrees of freedom**: in soft robots the degrees of freedom are more with respect to their rigid counterparts due to the diverse movements that can be exhibited by soft materials, like bending, stretching, twisting and compressing. This improved mobility gives great advantages because they have a greater range of motion and so are more adaptable to unpredictable environments.[67]

2. **Low component cost**: soft materials, such as elastomers and polymers, are usually less expensive than traditional rigid materials like metals and alloys, used for rigid robots. Also many soft robots are fabricated using simple and cheap manufacturing methods such as molding and 3D printing, which require less equipment and resources than for rigid robots.[49][2]

3. **Safety**: since usually the surface of soft robots is adequately soft and deformable, it's able to distribute forces over a large contact area eliminating interfacial stress concentrations (which for contact with human tissues may cause physical discomfort or injury)[42] this leads to safer human-robotic interaction in general [61]

4. **Compliance matching**: ability of the robot's flexible and deformable structure to readily change shape or deform in response to applied forces or stimuli to adapt to the environment. The main characteristic that influence the compliance is the Young's module, which is a useful measure for comparing the rigidity of different materials. Rigid robots which are composed usually of metal or plastics have a modulus greater than 10^9Pa, while most of the materials in natural organisms have a much lower Young's module ($10^2 - 10^6$). In order to avoid this great mismatch of rigidity, the soft robots are defined as systems that are capable of autonomous behaviour, and that are primarily composed of materials with moduli in the range of that of soft biological materials.[42]

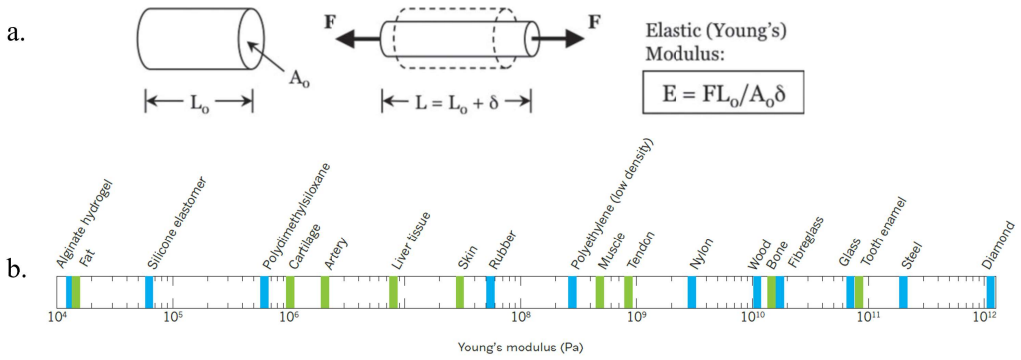


Figure 2: (a) The elastic (Young's) modulus scales with the ratio of the force F to the extension d of a prismatic bar with length L0 and cross-sectional area A0.[42] (b) Young's modulus for various materials.[67]

5. **Resilience**: soft robots are usually very resilient due to their flexible and deformable structure, which absorbs energy, mitigates damage, and enables adaptation to various conditions.[68]

6. **Adaptability**: soft robots offer high adaptability due to their flexible and deformable nature thus allowing them to conform to complex and dynamic environments. [74]

7. **Robustness**: ability to maintain functionality and performance despite variations in operating conditions, environmental factors or external disturbances.[17]

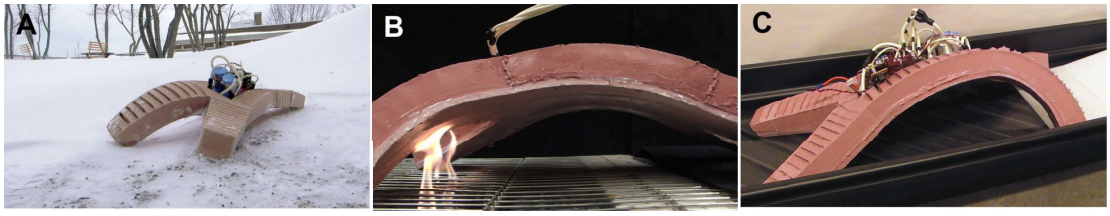


Figure 3: Example of a soft robot resilient to harsh conditions,
in particular: (A) a snow storm, (B) a fire and (C) water [68]

8. **Biocompatibility**: soft robots in general are not always biocompatible because it depends on the specific materials used, but it could be useful for in-vivo application to have biocompatibility. As efforts in synthetic biology and tissue engineering advance it's always more plausible to expect biocompatible technologies within soft robots.[42]

1.1.1 Applications

Soft robotics offer a wide range of applications: from soft grippers and artificial muscles to wearable devices and medical tools (Fig [**fig:applications**]).
With respect to the medical field and wearable applications their intrinsic safety and their minimal invasivity[34] allow for safe and biomechanically compatible interactions with humans, due to this it's spreading the study of soft robotics in the rehabilitation field with wearable applications (bio-inspired soft wearable robotic for rehabilitation[59][14], soft sensing suits for lower-limb measurement[47], soft system for simulation of cardiac actuation[64]).
For field exploration or disaster relief soft robots have the advantage of being able to navigate challenging terrains and the ability to fit into tight spaces by adapting their shape and locomotion strategy.
Miniature soft robots are a new field but it holds promise in for medical applications like drug-delivery and biopsy.[42]

Specific Applications of Soft Actuators from Universidade de Coimbra

The soft actuators developed in our lab exhibit unique properties that enable a range of innovative applications, particularly in the fields of rehabilitation and robotics.

- **Climbing Robot:**
  The soft actuators have been successfully integrated into a robotic system designed for tree climbing. This application utilizes the actuators' flexibility and adaptability, allowing the robot to navigate complex surfaces and climb with precision. The actuators function as the robot's legs, providing both grip and stability during ascension.

- **Soft Robotic Hand:**
  Another application involves using the soft actuators to mimic the functionality of human fingers in a soft robotic hand. In this design, three actuators are strategically placed to control individual fingers: one actuator for the thumb, one for the index finger, and another for the remaining three fingers. This configuration allows for a versatile range of movements and gripping capabilities, showcasing the actuators' potential in dexterous tasks.

- **Soft Gripper:**
  In a different design iteration, our soft actuators have been utilized in a soft gripper application. This design emphasizes the ability of the actuators to conform to various object shapes, allowing for gentle and effective gripping without damaging delicate items. The soft gripper exemplifies the advantages of using soft robotics in handling fragile objects in industrial and medical settings.

- **Rehabilitation:**
  The last promising application, which is for now just an idea, is in the field of rehabilitation. The inherent flexibility of the actuators allows for gentle movement of muscles undergoing rehabilitation, providing a slow and controlled motion that minimizes discomfort for the patient. This characteristic is particularly beneficial when working with sensitive areas, as the softness of the actuators contributes to greater comfort compared to traditional rigid devices. Additionally, the heat generated by the actuators can enhance therapeutic effects. Research has shown that localized heat application effectively reduces pain perception and promotes vasodilation, which increases blood flow in compromised areas, furthermore the heat can improve the viscoelastic properties of connective tissues, facilitating the rehabilitation process [54]. The recommended duration for applying these actuators to the skin is approximately 20 minutes, allowing for effective treatment without excessive evaporation of the working fluid.

Overall, the versatility and unique properties of this soft actuator open up numerous possibilities across various domains, enhancing their applicability in rehabilitation and robotic systems.

1.1.2 Challenges

Since the field of soft robotics is new, there are many new challenges to take into account and most of them are based on the fact that the research is still developing. Below are explained the most known challenges of the field.

- **Lack of simulation and analysis tools**

Simulating the dynamics of soft materials is difficult due to their numerous degrees of

freedom and non-linear effects due to their materials. An extensive computational process is then needed to be employed for an accurate simulation and even if the simulation appears correct predictions are usually difficult to match the reality due to the need for empirical calibration of coefficients.[39]

- **Lack of design automation tools**

Design automation is essential in soft robotics due to the complexity of the soft systems which limits human intuition regarding their behaviour of kinematics and dynamics. However the devolpment of design automation tolls requires new mathematical representations and accurate, fast simulators, which are both mostly lacking.[39]

- **Lack of soft actuation methods**

There are diferent actuation challenges which make untethered soft robots difficult to realize: electroactive polymers usually demand high voltage, whereas ionic polymer metal composites operate at lower voltages but are inefficient, slow and weak; additional pressure infrastructure is required for pneumatic actuation while shepe-memory wires necessitate high power.[39]

- **Lack of control authority**

The main challenge concerning the control is due to the infinite degrees of freedom that can be used to describe a soft robot due to their elastic nature and diverse range of movements.[67]
Traditional robotic control methods usually rely on precision actuation and control authority which are difficult to achieve with soft materials. Since impedence is reduced with soft materials, while lag times, deformations and vibrations is increased; the feedback control which assume high structural impedence to adjust velocities and positions becomes much harder. The new solutions will then require more sensing and modeling, as well as the development of new control methods and design paradigms.[39]

- **Primitive fabrication methods, modularity, and standards**

To replicate the shape and the rigidity-tunable properties of natural muscle tissue new classes of electrically and chemically powered soft actuators have been developed [42].
The main source of problem regarding the fabrication is related to the fact that unlike with rigid robots, standardized parts to assemble don't exist, so each project needs to start both design and manufacturing from scratch; this creates difficulties in the optimization and in the knowledge transfer. [39]

1.2 Objectives

The primary objective of this work is to design and implement control strategies for an untethered soft actuator, specifically a thermo-electric artificial muscle. The focus is on developing various control approaches and evaluating their performance through both simulation (in Simulink) and real-world experimentation.

To achieve this, the following tasks were undertaken:

1. **Characterization of the Soft Actuator:** Through extensive experimentation, the behavior and dynamic properties of the thermo-electric soft actuator were studied and this phase helped in gaining insights into the actuator's response characteristics, which were then used to inform the design of suitable controllers.

2. **Controller Design:** Multiple controllers were designed, taking into account the nonlinear and time-dependent behavior of the soft actuator; specifically both traditional and advanced control strategies were explored, with the goal of optimizing the actuator's performance in different scenarios.

3. **Simulation and Real-World Implementation:** The designed controllers were first implemented and tested in a simulated environment using Simulink, this allowed for a preliminary assessment of the control algorithms in a controlled, repeatable setting. After validation in simulation, the controllers were then implemented on the physical actuator to assess their real-world performance.

4. **Comparative Analysis:** The performance of the various controllers was compared based on several criteria, such as response time, accuracy, and robustness; both simulated and real-world results were analyzed to provide a comprehensive understanding of the strengths and limitations of each control approach.

In conclusion, the project aims to bridge the gap between simulation and practical implementation by validating the control strategies in both environments, ensuring their applicability for untethered, soft robotic systems.

1.3 Thesis outline

The thesis is divided in five chapters, the first being the introduction. The second one will present the state of the art of both the types of soft actuators and the different control strategies utilized. The materials and methods taken into account for the experiment are expressed in chapter three; then in the fourth chapter are explained the results of the experiments conducted along with a discussion about them. Finally the last chapter is dedicated to the final remarks.
The sequence of steps followed is done following the design flow as follows:

- **Model in the Loop Testing**: simulation helps in refining both models and evaluate design alternatives. Both the controller and the plant in this part are in the same simulation tool (Simulink)

    1. System requirements
    2. System design: model of the Plant (system to be controlled) and of the Controller (algorithm to control the Plant)
    3. Software design

- **Optimization and Code Generation**: translates the model into code.

- **Software in the Loop Testing**: simulation to verify if the code is consistent. The implementation is co-simulated with the plant model to test its correctness. It is still executed on a PC.
  Part of the code exists in native simulation tool (Simulink) and part as executable C-code. Good for testing controller implementations in C-code.

    1. Software Design
    2. Coding
    3. Software integration

- **Processor in the Loop Testing**: the implementation is deployed on the target hardware and is co-simulated with the plant model to test its correctness. The implementation runs on target hardware (no real time yet).
  Part of the model exists in native simulation tool (Simulink) and part as executable C-code run on target hardware or rapid prototyping hw. Good for testing controller implementations in C-code.

    1. Software integration
    2. Hardware integration

- **Hardware in the Loop Testing**: integration of the item in the physical system. The implementation is co-simulated with the plant model to test its correctness. The implementation runs in target hw, the plant model on rapid prototyping hw (real time).
  Part of the model runs in a real-time simulator, and part may exist as physical hardware. Good for testing interactions with hw and real-time performance.

    1. Hardware integration
    2. Actuators integration and Calibration

2 State of the Art

2.1 Soft Actuators

A soft actuator is an actuator whose physical form and flexible material allow for actuation (axial, radial, torsion, bending) even under physical perturbations (bending, pinching, or pulling) when energy input is applied. [25]
Below a classification of the most common types of soft actuators. [77]

2.1.1 Piezoelectric Actuators

Piezoelectric actuators utilize the piezoelectric effect to transfer input electric energy into force or displacement in order to deform the material to produce precise movements or apply the force. In general the piezoelectric effect [60] refers to the ability of a material, usually ceramic, to generate electrical energy in response to mechanical stress, while the inverse piezoelectric effect does the opposite.[43]

Advantages: compact in size, flexible design[21], high displacement accuracy, large generation force, fast response time.[43]

Disadvantages: small range of motion, piezoelectric material is brittle, high voltage requirements (which can cause concerns with safety). [77]

Applications: nanopositioning, precision machining, adaptive optics and medical devices.

2.1.2 EAP actuators

EAP stands for Electroactive Polymers, these type of actuators use this polymers which change shape or size in response to an electric field. There are two major groups of EAP actuators: with ionic EAP materials, whose actuation is involved with diffusion of ions, and with field-activated EAP materials, which are driven by Coulomb forces and may require high voltages.[5] One of the most famous type of EAP actuator is the dielectric elastomer actuator (DEA), it consist of a soft elastomeric polymer film coated with compliant electrodes. When a voltage is applied between the electrodes they move closer to each other thanks to a compressive Maxwell stress.[15]

Advantages: ability to mimic biological muscles[4], mechanical flexibility, low density, low power consuption, easy to process[5], large strain and high bandwidth.[77]

Disadvantages: challenging to operate at practical voltages, difficulty in modelling and control and low physical robustness. [77]

Applications: Robotic arms and grippers, biomimetic robots, humanoid robots, soft robots, robotic fishes and vehicles.

2.1.3 SMA and SMP Actuators

Shape Memory Alloy (SMA) actuators use special metal alloys to produce mechanical motion or force. They have the instrinsic ability to recover their original shape when subjected to a temperature higher than a predefined treshold.
Similarly, Shape-Memory Polymers (SMP) have the same ability to recover their initial state after deformation but with light, magnetic field, chemical reaction or temperature variation.

Advantages: lightweight, remotely controllable, employ low voltages.[72]

Disadvantages
SMA: need high currents, highly non-linear and therefore difficult to control.[72]
SMP: response time can be very long.[72]

Applications
SMA: medical robots, self-reconfigurable robots, biomimetic robots, robotic hands, manipulators, and exoskeletons.
SMP: biomedical devices and robotic origamis.

2.1.4 Twisted String Actuators

Twisted string actuators (TSAs) convert the twisting rotational motion into linear motion. The actuators consist of a string of fibers able to contract or expand in response to a force or a torque.

Advantages: scalability, high energy efficiency, large linear strain and stress outputs, the strings are stiff and strong.[8]

Disadvantages: the strings are not compliant and the precise control relies mostly on external position or force sensors.[8]

Applications: soft robotics, prosthetics, and haptic interfaces.

2.1.5 Soft Fluidic Actuators

The soft fluidic actuators are the most prevalent type of soft actuators, they work by controlling the fluid pressure inside the hollow channels of their body.[34]
The most important SFAs are pneumatically driven and are called PAMs (Pneumatically Artificial Muscles), they convert energy from compressed air to mechanical motion.[77]
Of the PAMs family the most studied and cited is the McKibben muscle, it was developed

in the 1950's and are currently commercialized.[13] They are constructed by coaxially locating a rubber tube within a woven sheath. The rubber tube creates an airtiht bladder whereas the woven sheath protects the bladder and converts its inflation into mechanical work.[77]

Advantages: highly compliant, easy to fabricate, can provide large deformations [34], have a simple and relatively high-performance design[50], safe for use in direct contact with humans.

Disadvantages: actuation is typically slow, inefficient and not very precise [34] and usually they require bulky peripheral components to operate. [50]

Applications: manipulators and grippers, biomimetic robots, wearable and assistive robots

2.1.6 SCP Actuators

The SuperCoiled Polymer actuators are constructed by twisting polymer fibers or filaments like carbon nanotube yarns [37], nylon fishing lines and sewing threads. The polymer threads contain long polymer chains aligned with the thread axis; under thermal expansion the diameter of the helices grows in diameter actuating also a change in twist. [76]

Advantages: large actuation range and significant mechanical power. [77]

Disadvantages: slow performance, small forces, difficult to obtain an accurate modeling and control, limited lifetime. [77]

Applications: robotic fingers, hands and arms

2.1.7 HASEL actuators

Hydraulically amplified self-healing electrostatic (HASEL) actuators are an hybrid of DEAs and SFAs and are therefore classified as electroelastic and thermoplastic actuators. They use both the electro-static and the hydraulic forces and they can actuate in different modes (expansion, contraction, rotation), they can self-sense their deformation and can electrically self-heal after a dielectric breakdown.[34]

Advantages: linearly contract on activation without stacks, pre-stretch or frames, high force production and scalability, inexpensive materials, simpler kinematics (with respect to other soft actuators).[31]

Disadvantages: limited operation lifetime, require high voltage.[31]

Applications: artificial muscles

2.2 Modeling for Soft Actuators

Soft actuators, which are a key component of soft robotics, present significant challenges in terms of modeling, planning, and control and unlike traditional rigid-body robots, which can be modeled with well-established kinematic and dynamic principles, soft actuators exhibit highly nonlinear and continuous deformation due to their intrinsic material properties. This characteristic makes it difficult to derive accurate mathematical models and control strategies, furthermore the modeling of soft actuators is complicated by the fact that they are often under-actuated and possess many passive degrees of freedom, especially when actuated by low-pressure fluids [67].
The deformation of soft actuators is typically continuous and distributed throughout their structure, leading to a high-dimensional state space that is difficult to capture using traditional modeling techniques. Moreover, the input fluid power may be insufficient to counteract external forces such as gravitational loads, leading to complex, non-linear behaviour that standard control algorithms difficultly can handle.
Several modeling approaches have been proposed in the literature to address these challenges: these approaches vary in complexity and the degree of physical insight they provide for the behaviour of soft actuators. The following sections describe the most common methods used to model soft actuators.

- **Cosserat Rod Theory**

  One of the most widely used methods to model the continuous deformation of soft actuators is *Cosserat Rod Theory* [9]. This theory generalizes the classical Euler-Bernoulli beam theory to allow for large deformations and bending in 3D space: Cosserat rods are modeled as one-dimensional, capable of capturing both bending and twisting motions, making them highly suitable for soft robotic structures, which often involve complex, flexible deformations.

  In the context of soft actuators, Cosserat Rod Theory provides a robust framework to model the kinematics and dynamics of long, flexible structures such as soft robotic arms or artificial muscles. By treating the soft actuator as a continuum body with distributed mass and elasticity, this approach allows for a more accurate description of the mechanical behaviour. However, the complexity of solving the resulting partial differential equations (PDEs) often necessitates numerical methods, such as finite element analysis (FEA), in order to obtain practical solutions.

- **Koopman Operator Theory**

  Another promising approach for modeling soft actuators is based on *Koopman Operator Theory* [53]. Koopman models offer a data-driven approach to capture the non-linear dynamics of soft actuators by lifting the system's dynamics into a higher-dimensional space where they can be treated linearly; this linear embedding allows

the complex, non-linear behaviour of soft robots to be described using a set of linear equations, which are easier to handle for control and optimization purposes.

Koopman operators are especially useful when combined with machine learning techniques, as they can be learned from empirical data. This data-driven approach is particularly well-suited to soft actuators, whose deformation is difficult to model analytically due to material non-linearity and complex geometries. By leveraging Koopman theory, the system dynamics can be learned directly from experimental data, allowing for improved performance in control and prediction tasks.

- **Long-Short Term Memory (LSTM) Networks**

  Machine learning approaches, particularly *Long-Short Term Memory (LSTM)* networks, have also been used to model soft actuators [10]. LSTMs are a type of recurrent neural network (RNN) capable of learning long-term dependencies in time-series data. For soft actuators, which exhibit dynamic behaviours with time-varying inputs and outputs, LSTMs can capture the actuator's non-linear temporal dynamics and provide accurate predictions of the future states.

  LSTMs are particularly advantageous for soft actuator modeling due to their ability to learn from experimental data, even in the presence of noise or incomplete information. Once trained, these models can predict the actuator's response to various inputs and external conditions, making them valuable tools for both control and planning. However, the quality of the model heavily depends on the amount and diversity of the training data, and the training process can be computationally intensive.

- **N-Link Pendulum Model**

  A more simplified approach for modeling the kinematics of soft actuators is the *N-Link Pendulum Model* [27]. This model discretizes a soft actuator into a series of rigid links connected by flexible joints, effectively treating the actuator as a chain of pendulums. Each link in the model represents a small segment of the actuator, and the overall deformation is captured by the relative motion between the links.

  Although this model does not fully capture the continuous deformation of a soft actuator, it provides a computationally efficient approximation that is often sufficient for control purposes. The N-Link Pendulum Model is particularly useful for soft actuators with segmental designs or for those operating in constrained environments where the deformation is limited to certain degrees of freedom.

- **Finite Element Analysis (FEA)**

  Finite Element Analysis (FEA) is a numerical technique often employed for the detailed modeling of soft actuators. This method discretizes the soft actuator's

geometry into small, finite elements, allowing the continuous deformation to be approximated by solving the equations of motion for each element. FEA is particularly well-suited to capturing the highly non-linear, elastic behaviour of soft actuators, making it a valuable tool for both design and analysis.

FEA is widely used in the design phase to predict how soft actuators will behave under various loading conditions, including bending, stretching, and twisting; but while it provides accurate results, the computational cost can be high, particularly for real-time control applications. As a result, FEA is often used in conjunction with other modeling methods that are more suitable for real-time implementation.

In summary, the modeling of soft actuators remains an open research problem, with no single approach offering a complete solution for all types of soft robots, each of the modeling techniques discussed has both advantages and limitations. For example, Cosserat Rod Theory provides detailed insights into the mechanical behaviour of continuum robots, while Koopman operators and LSTMs offer data-driven methods that are more adaptable to real-world uncertainties. The choice of model depends on the specific requirements of the application, including the desired balance between accuracy and computational efficiency.
In the end the type of modeling we chose to follow for the soft actuator was to derive the fundamental physics of the actuator's components their general behaviour and apply that in the modeling.

2.3 Controls for soft actuators

There are four different approaches used to build a controller for a soft acuator: two of them are based on a model of the actuator while the other two treat the model as a black-box obtaining a simpler system but also a less accurate one. The other diferentiation is if the controller are kinematics (static) or dynamic, the static are the most researched because easier, while the dynamic are more difficult are therefore still unexplored.

2.3.1 Model-based kinematic controllers

The kinematic models can be developed adopting the steady-state assumption that the configuration of the soft manipulator can be defined by a low-dimensional state-space representation. The *constant curvature* (CC) approximation[66] is the most used model and it reduces an infinite dimensional structure in 3D (three variables) as sufficient parameters for a continuum-soft module. This is a good approximation just in the case the manipulator is uniform and symmetric, torsional effects are minimal and external loading effects are negligible.

Other advanced modeling approaches used for soft actuators and robots are Piecewise Constant Curvature (PCC) models[29], beam theory[23], and Cosserat rod theory[70]. These methods offer a more sophisticated representations of the mechanical behavior, accounting for factors like multi-sectional manipulators, large-deflection dynamics, and complex continuum bodies but they also often come with computational burdens, platform specificity and demand for a lot of sensory data.

2.3.2 Model-free kinematic controller

Model-free approaches offer the advantage of not requiring predefined parameters in configuration or joint spaces, making them adaptable to various manipulator shapes. They are mostly used in highly nonlinear, non-uniform systems or in systems operating in unstructured environments where modeling is challenging. [59]
However, for well-behaved manipulators in known environments, model-based controllers remain more accurate and reliable also because the black-box nature of the model-free approaches complicates stability analysis and convergence proofs. Moreover static/kinematic controllers assume minimal dynamic coupling between sections, limiting accurate and fast motion; therefore dynamic controllers are crucial for faster, dexterous, and smoother tracking, especially when dealing with coupling effects.[22]

2.3.3 Model-based dynamic controllers

Dynamic controllers are vital for industrial applications since they prioritize accuracy, time and cost. Direct mapping dynamic models offer optimal performance but are limited since they mostly focuse on joint space control; for example Model Predictive Controllers (MPCs) show promise but face many computational challenges. Moving to a non-steady-state approach, the high dimensionality of the robots create a lot of problems. Dynamic controller are still in their nascent stage, and for this reason the research on this topic is still sparse, an alternative could be machine learning-based approaches which offer potential for dynamic control.[22]

2.3.4 Model-free dynamic controllers

Lastly the model-free dynamic controller which is the least studied one: it has different advantages such as being simpler to develop a dynamic model and being virtually platform-independent, as well as some disadvantages like training time and impossibility of reaching the same stability of a classic controller.

3 Materials and Methods

3.1 Thermo-electric actuator

The actuator on which this thesis is based has been designed by Diogo Fonseca a PhD student of the university of Coimbra. The soft-actuator is thermo-electric and in particular is liquid-gas phase trasition actuator and is inpired in the design on the McKibben Pneumatic Artificial Muscles (PAM), specifically on the flexible bladder which is a cylindrical tube shrouded by a braided sleeve made of non-extensible threads. In the PAM the inflation of the inner bladder causes an increase in volume, which is constrained by the sleeve leading to longitudinal contraction or extension, depending on the specific characteristics of the braid used. Instead in the actuator we are using adds an electric heating element and a work fluid to the PAM. The operating principle is: use electric energy to heat and boil the work fluid in order to provide the pressurization necessary for the desired deformation. This will be referred to as a Phase Change Artificial Muscle (PCAM).
Each actuator is composed of 6 main elements:

1. End terminals
2. Heating element
3. Working fluid
4. Inflatable bladder
5. Braided sleeve
6. Pressure Sensor

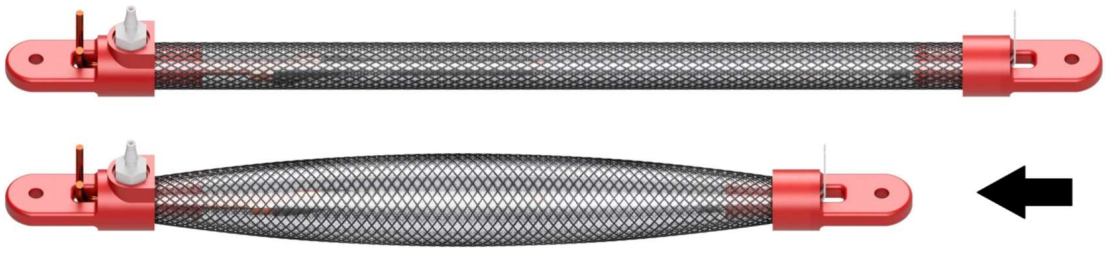


Figure 4: Electrically powered Phase Change Artificial Muscle (PCAM)

3.1.1 End Terminals

The end terminals of the actuator are designed to route two electrical connectors for the heating element through a single terminal, which helps keep the heating coil stable and centered. This design improves durability by reducing the risk of fatigue and minimizing the accumulation of aluminium oxide particles (Al_2O_3) on the FeCrAl heating element, which can occur due to mechanical and thermal stresses.
The extremities of the actuator are produced using 3D printing technology, specifically with PET/G material for durability and ability to form airtight and watertight seals. These components are printed using a Prusa 3D printer, with designs created in Autodesk Inventor by Diogo Fonseca [18]. Figures 5, 6, and 7 show the final designs of the actuator extremities.

Figure 5: New design of actuator extremities.

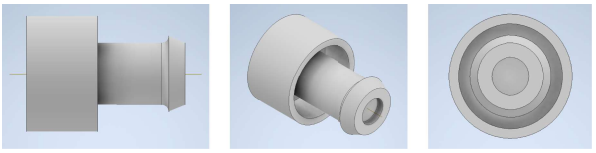


Figure 6: Actuator extremities in roll configuration.

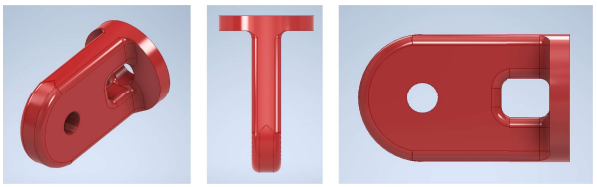


Figure 7: Detailed view of the printed actuator extremities.

The printing process takes approximately 16 minutes per piece at a temperature of 265°C. While PET/G was used for the prototypes, alternative materials such as nylon reinforced with carbon fiber (Fiberology Nylon PA12+CF15) are being considered for future designs to enhance the structural integrity and long-term durability of the actuator components.

3.1.2 Heating Elements

The heating element used in the actuator consists of a FeCrAl wire with a diameter of 0.35 mm, arranged in a coil configuration. FeCrAl is chosen due to its higher electrical resistivity compared to other commonly used alloys such as NiCr, NiFe, or CuNi, this property allows the actuator to operate with lower currents for a given power output and heating element geometry, making it more efficient in terms of energy use.
Although NiCr alloys offer superior wet corrosion resistance, FeCrAl provides a key advantage in the formation of aluminum oxide (Al_2O_3) on its surface. This oxide layer adheres more effectively to the alloy and serves as a superior electrical insulator compared to the chromium oxide (Cr_2O_3) formed on NiCr alloys. The insulating properties of Al_2O_3 contribute to the thermal efficiency and durability of the heating element in repeated heating and cooling cycles.[18]

3.1.3 Working Fluid

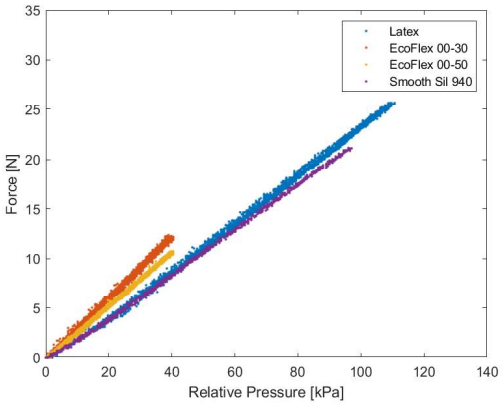
The working fluid inside the actuator is water, which undergoes a process known as pool boiling. Pool boiling refers to a convection heat transfer mechanism where the liquid (in this case, water) is in direct contact with a hot surface, such as the heating coil, whose temperature exceeds the saturation temperature of the liquid (T_{sat}).
Vapor bubbles nucleate at the solid-liquid interface, growing in size until they detach and rise towards the surface. These bubbles, typically not in thermodynamic equilibrium with the surrounding liquid, facilitate heat transfer between the liquid and vapor phases. In cases where the temperature of the adjacent liquid is significantly lower than its saturation temperature, the vapor bubbles collapse before reaching the surface, a phenomenon known as subcooled pool boiling. Conversely, when the liquid's temperature approaches its T_{sat}, vapor bubbles rise fully to the surface, a process referred to as saturated pool boiling.

Figure 8: Heating of the coil under water demonstrating subcooling and pool boiling. [19]

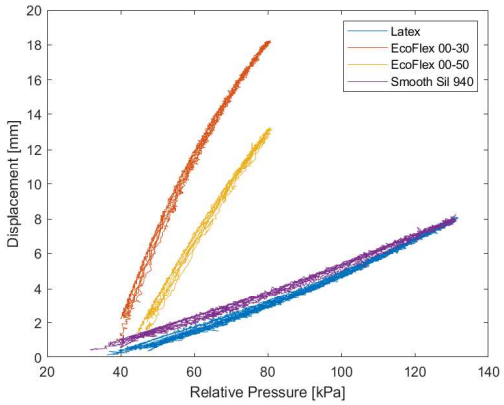
The collapse of vapor bubbles during subcooled pool boiling shares similarities with cavitation, although the underlying causes differ: subcooled pool boiling results from localized temperature increases, while cavitation is caused by localized pressure drops. In both cases, the collapse of bubbles generates high-frequency shockwaves that propagate through the liquid medium and these shockwaves can have a significant impact on the actuator's internal components, particularly the heating element and sensors.

3.1.4 Inflatable Bladder

Four different materials were selected and tested for use in the inflatable bladder: natural rubber latex (6x9 mm Latex Amber Tubing from StonyLab) and three types of platinum-catalyzed two-part silicone rubbers (EcoFlex 00-30, EcoFlex 00-50, and Smooth-Sil 940, all from Smooth-On). To facilitate meaningful comparisons, all silicone bladders were fabricated to match the dimensions of the latex bladder, which came pre-formed as a 6x9 mm tube.
All the inflatable bladders had a cylindrical shape with internal diameters of 6 mm and external diameters of 9 mm. The tube length was set to a minimum of 150 mm (+20 mm for fitting to the end terminals) to ensure that the heating element remains fully submerged in liquid when the actuator is vertically mounted and operating at its maximum design pressure of 230 kPa at 10% strain. Longer tubes allow for greater linear displacement, while shorter lengths could expose the heating element to the gaseous phase, leading to an increase in surface temperature due to the higher thermal resistance of solid-gas convection. Although early latex-based prototypes were tested in 100 mm and 200 mm variants, the standard length chosen was 150 mm.



(a) *Isometric testing comparing different mold types for the actuator's tubes*



(b) *Isotonic testing with a 1 kg weight comparing different molds for the actuator's tubes*

Silicone Tubes

For the fabrication of silicone tubes, 20 grams of silicone material were used per tube. The material selected for this process was EcoFlex 00-30, as it was found to offer the greatest displacement, despite being less optimal in terms of the force it could withstand. The decision to prioritize displacement over force was based on the specific application requirements.
The silicone mixture is placed in a vacuum chamber, where its volume expands significantly before boiling and settling after approximately 10 minutes. Once the air has been extracted from the silicone, the mixture is poured into the molds, as shown in Figure 10. The molds are then returned to the vacuum chamber to remove any remaining air bubbles from the silicone. After the bubbling stops, and all air has been evacuated, a metal rod is inserted into the silicone-filled mold to create the hollow center of the tube.

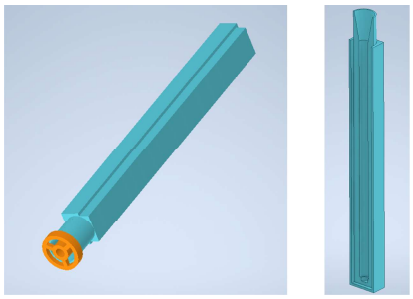


Figure 10: Silicone mold used for creating the inflatable bladder tubes.

After the silicone has fully solidified (which requires at least one day), the tube is cut to a length of 17 cm (in its un-stretched state) and is then securely glued to the end terminals of the actuator. During testing, we observed that excessive heating of the tube could weaken the glue's bond to the end terminals, potentially leading to the actuator bursting, as shown in Fig. 11, which can be a possible weakness of the design.

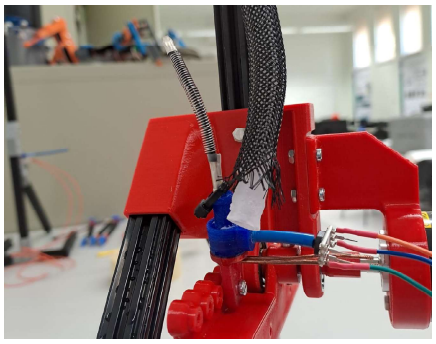


Figure 11: Bursting of the actuator after excessive heating.

3.1.5 Braided Sleeve

A braided PET sleeve (from SES Sterling) was selected for the actuator due to its low internal friction and thermal conductivity, making the device safe to touch even at higher temperatures. However, this comes at the cost of slower relaxation speeds. Most commercially available braided sleeves follow one of four common braiding patterns. Among these, biaxial braids were chosen, with the 3x3 braid considered the most efficient due to its lower energy dissipation from yarn friction.
The extensibility of the actuator is influenced by the braid's geometry. The total length of the braid (L) can be approximated as:

$$L = b \times \cos(\theta) \tag{1}$$

where b is the length of the yarn, and θ is the angle between the yarn and the braid's longitudinal axis.
The force exerted by the braid under internal pressure (P) is expressed by:

$$F = \frac{Pb^2}{4\pi n^2}(3 \times \cos^2(\theta) - 1) \tag{2}$$

where n represents the number of yarn turns around the braid's axis.
The "neutral angle," where the longitudinal force is zero, is approximately 54.7°, representing the upper limit of the braid angle during contraction.
The braid's ability to contract is influenced by the Cover Factor (CF), which can be calculated as:

$$CF = 1 - (1 - \frac{n \times W_y \times N_y}{b \times \sin(\theta) \times \cos(\theta)})^2 \tag{3}$$

where W_y is the yarn width and N_y is the number of yarns. A lower CF value indicates a more contractible braid. Braids with lower N_y or narrower yarns are more suitable for actuator applications, as they allow for greater contraction. However, as the braid angle approaches 45°, the risk of the inflatable bladder bulging and rupturing increases due to exposed areas in the braid.

3.1.6 Pressure Sensor

A pressure sensor (ABPDANV060PGAA5 from Honeywell) was connected to each actuator's static port via a soft polyurethane pneumatic tube (TUS0425 from SMC). To minimize feedback lag, the static pressure port was positioned on the lower side of the actuator, ensuring that pressure transmission occurred through the liquid phase, which has negligible compressibility and reduces pressure loss. In some prototypes, a temperature sensor (DS18B20 from Maxim Integrated) was also installed for additional monitoring.

The actuator is powered electrically, with a power supply connected to the coil inside the actuator. As the coil heats up, the surrounding water reaches boiling, causing the silicone tube to contract. This results in either displacement (in isotonic tests) or force generation (in isometric tests). Upon removing the current, the coil and water gradually cool, allowing the actuator to relax through natural air cooling.

3.1.7 Assembly

This section outlines the entire assembly process of the soft actuator, detailing the steps involved in preparing the heating coil, connecting cables, and integrating protective components to ensure proper function.
The heating coil is formed from a spring cut to a length of 58 mm, while the copper cables are stripped of their silicone insulation and cut to a length of 75 mm, providing a resistance of 25 Ω. The cable ends are soldered and covered with heat shrink tubing (15–13 mm) to protect the silicone from potential damage caused by the metal edges.
Additional metal pieces are clamped to another set of cables, either 115 mm in length (providing 25 Ω) or 45 mm (for 8 Ω). A second coil is cut to 58 mm and stretched to 100 mm, giving 25 Ω (or 8 Ω in a shorter configuration). Once the coil and cables are assembled, a silicone tube is placed around the setup, followed by a braided PET sleeve (8 mm) that acts as thermal insulation, limiting heat transfer from the interior to the exterior.

Figure 12: Actuator half assembly: lower-end terminal, static port, electric connectors, and heating element [18]

Several factors influence the actuator's real-life performance:

- The amount of water filled into the soft actuators.
- The degree to which the actuator is stretched and straightened during testing.
- Variations in production, since all actuators are manually fabricated.

Hand fabrication introduces variability in the results, such as differences in coil or wire lengths, which can alter the actuator's resistance and, consequently, the current passing through the coil.

3.2 Modeling

A model is a crucial element in understanding and controlling a system. It serves various purposes such as prediction, simulation, filtering, decision-making, and *control*. In the context of soft actuators, the first step involved creating a detailed model that accurately represents the physical principles governing their operation.
Model-driven engineering revolves around several key concepts:

- Building a system model that captures relevant features.
- Using simulations to evaluate the system's properties and behaviors.

The primary data for this model is derived from the work of Chou and Hannaford [13]. We consider an ideal McKibben-style pneumatic artificial muscle (PAM) characterized by the following assumptions:

- Zero static and dynamic friction.
- Inflatable bladder composed of an infinitely soft material (zero Young's modulus).
- Inflatable bladder with zero thickness.
- Braided sleeve constructed from infinitely rigid strands.

As described by Chou and Hannaford [13], the output force (F) of this actuator can be calculated using Eq. (4):

$$F = \frac{Pb^2}{4\pi n^2}\left(3\cos^2(\theta) - 1\right) \tag{4}$$

where:

- P [Pa] is the relative internal pressure.
- b [m] is the fixed length of the braid yarns.
- n is the number of turns each yarn makes around the braid's longitudinal axis.
- θ is the variable braid angle.

The braid angle is a function of the braid's current length, as described by Eq. (5):

$$L = b\cos(\theta) \tag{5}$$

By combining Eq. (4) and Eq. (5), we can derive Eq. (6) for the ideal pressure (P_{ideal}):

$$P_{ideal} = F\left(\frac{4\pi n^2}{3L^2 - b^2}\right) \tag{6}$$

3.2.1 Accurate Model

The accurate model developed here aims to simulate the working principles specific to the soft actuators described earlier. The model's implementation is based on the physical principles governing the actuator's behavior.
The model outputs:

- **gpress**: Gauge pressure [Pa].
- **state**: State variables (to be defined or describe them).

Inputs to the model include:

- **power**: Electric power supplied by the generator [W].
- **strain**: Current strain of the actuator [%].
- **force**: (Consider including this variable for visualization?) [N].

```
function [gpress, state] = ptam(power, strain, state)
```

Key formulas in the model are inspired by the McKibben artificial muscle [13]. The model function `ptam` calculates the gauge pressure `gpress` and updates the state variables based on the input parameters.
The parameters used in the model are all contained in a parameter vector called `parma` and are defined as follows:

- `precision`: Pressure precision in each iteration.
- `l0`: Initial length of the actuator [m].
- `d0`: Initial diameter of the actuator [m].
- `theta0`: Initial braid angle of the actuator [deg].
- `mass`: Mass of the fluid inside the actuator [kg].
- `mass_coil`: Mass of the heating coil [kg]. (Note: multiplied by 4 for thermal mass compensation)
- `area_coil`: Surface area of the heating coil [m^2].
- `cp_liq`: Specific heat capacity of water [J/(kg*K)].
- `cp_coil`: Specific heat capacity of the heating coil [J/(kg*K)].
- `h1`: Convection heat transfer coefficient [W/(m^2*K)].

- `h2`: Liquid-gas interface heat transfer coefficient [W/(m^2*K)].
- `h3`: Metal-to-liquid heat transfer coefficient [W/(m^2*K)].
- `tamb`: Ambient temperature [K].
- `bp`: Boiling point of the fluid at atmospheric pressure [K].
- `L`: Latent heat of vaporization of the fluid [J/kg].
- `k`: Yarn rigidity parameter [Pa^{-1}].
- `dead_vol`: Dead volume inside the actuator [m^3].
- `patm`: Atmospheric pressure [Pa].
- `simstep`: Simulation time step [s].

This parameter vector, `param`, encapsulates all necessary constants and initial conditions for the model simulation.

```
param.precision = 100;
param.l0 = 0.17;
param.d0 = 0.01;
param.theta0 = 40;
param.mass = 0.002;
param.mass_coil = 3e-3;
param.area_coil = 5.4978e-04;
param.cp_liq = 4186;
param.cp_coil = 500;
param.h1 = 25;
param.h2 = 500;
param.h3 = 1500;
param.tamb = 25 + 273.15;
param.bp = 100 + 273.15;
param.L = 2.26e6;
param.k = 0;
param.dead_vol = 2.4008e-07;
param.patm = 101330;
param.simstep = 1e-3;
```

This modeling approach integrates theoretical insights with practical considerations, aiming to accurately replicate the behavior of soft actuators under various conditions.

Outer Surface Area Calculation

The outer surface area of a McKibben actuator is a crucial parameter for determining its physical properties and behavior under various conditions. Accurate modeling of the actuator's surface area enables a deeper understanding of its mechanical and physical

characteristics, particularly in response to external stimuli and environmental changes. This section details the calculation of the actuator's surface area, which is implemented within a function based on key geometric and strain parameters.
The surface area calculation depends on the following parameters:

- *Longitudinal Contractile Strain* (`strain`): This parameter represents the actuator's elongation or contraction along its longitudinal axis, expressed as a percentage. Positive values indicate contraction. [Unit: %]
- *Initial Length of Braided Sleeve* (`l0`): This is the unstrained length of the actuator's braided sleeve in meters. [Unit: m]
- *Initial Diameter* (`d0`): This represents the initial diameter of the actuator in meters. [Unit: m]
- *Initial Braid Angle* (`theta0`): This parameter signifies the angle between the yarn direction and the longitudinal axis of the actuator in its undeformed state, expressed in degrees. [Unit: deg]

The calculated surface area, denoted by `area`, is used within the model to analyze various aspects of the actuator's performance under different operating conditions.
The calculation of the surface area involves a series of geometric transformations that consider the initial and deformed states of the actuator. The steps involved in this computation are outlined below.

1. *Yarn Length Calculation*
   The initial yarn length, denoted as `yarn_length`, is computed based on the initial length of the actuator and the initial braid angle. This calculation assumes that the yarn's length remains constant throughout the actuator's deformation:
   $$\texttt{yarn\_length} = \frac{l_0}{\cos(\theta_0)} \tag{7}$$
   where:
   - l_0 is the initial length of the actuator's braided sleeve.
   - θ_0 is the initial braid angle of the actuator.
2. *Number of Turns Calculation*
   The number of turns around the braid, denoted as `num_turns`, is determined using the initial yarn length, the initial diameter, and the initial braid angle:
   $$\texttt{num\_turns} = \frac{\sin(\theta_0) \cdot \texttt{yarn\_length}}{\pi d_0} \tag{8}$$
   where:

- d_0 is the initial diameter of the actuator.

3. *Current Length Calculation*
   The current length of the actuator, denoted as l_1, after deformation is determined by considering the initial length and the applied strain:

$$l_1 = l_0 \left(1 - \frac{\text{strain}}{100}\right) \tag{9}$$

   where:

   - `strain` is the longitudinal contractile strain expressed as a percentage.

4. *Current Braid Angle Calculation*
   The current braid angle, denoted as θ_1, is recalculated considering the geometric relationship between the initial and current lengths:

$$\theta_1 = \cos^{-1}\left(\frac{l_1 \cos(\theta_0)}{l_0}\right) \tag{10}$$

   This equation captures the dependency of the braid angle on the contraction of the actuator.

5. *Current Perimeter Calculation*
   The current perimeter of the braid, denoted as `curr_perimeters`, is determined using the current yarn length and the current braid angle:

$$\texttt{curr\_perimeters} = \sin(\theta_1) \cdot \texttt{yarn\_length} \tag{11}$$

   The single yarn perimeter, `single_perimeter`, is calculated by dividing the current perimeter by the number of turns:

$$\texttt{single\_perimeter} = \frac{\texttt{curr\_perimeters}}{\texttt{num\_turns}} \tag{12}$$

6. *Outer Surface Area Calculation*
   Finally, the outer surface area, `area`, of the actuator is determined by multiplying the single yarn perimeter by the current length of the actuator:

$$\texttt{area} = \texttt{single\_perimeter} \cdot l_1 \tag{13}$$

   This calculated surface area is crucial for modeling the actuator's performance as it directly influences the mechanical response under various loads and pressures.

The function `surface_area` provides a comprehensive method to calculate the outer surface area of a McKibben actuator, accounting for changes in strain and braid angle. Understanding these geometric transformations allows for more accurate modeling of the actuator's physical properties and its response to external stimuli. This approach is fundamental in the design and optimization of soft actuators for a wide range of applications. The function `surface_area` is called as follows:

```
surfarea = surface_area(strain, param.l0, param.d0, param.theta0);
cross_area = pi * (state(8) / 2)^2;
```

While is defined as:

```
function area = surface_area(strain, l0, d0, theta0)
  yarn_length = l0 / cosd(theta0);
  init_perimeters = sind(theta0) * yarn_length;
  num_turns = init_perimeters / (pi * d0);
  l1 = l0 * (1 - (strain / 100));
  theta1 = acosd((l1 * cosd(theta0)) / l0);
  curr_perimeters = sind(theta1) * yarn_length;
  single_perimeter = curr_perimeters / num_turns;
  area = single_perimeter * l1;
end
```

Volume Calculation of a McKibben Actuator

The volume of a McKibben actuator is a critical parameter that influences its mechanical performance and internal pressure dynamics. Accurately calculating the volume allows for a better understanding of the actuator's response to different operating conditions, helping in the design and optimization of soft actuators. This section details the volume calculation, implemented within a function called `vol_gen`, which considers the geometric properties and deformation of the actuator.

The volume calculation depends on the following parameters:

- *Longitudinal Contractile Strain* (`strain`): This parameter represents the actuator's elongation or contraction along its longitudinal axis, expressed as a percentage. Positive values indicate contraction. [Unit: %]
- *Yarn Rigidity* (`k`): Describes the rigidity of the yarn; `k` is set to 0 for rigid yarns.
- *Initial Length of Braided Sleeve* (`l0`): This is the unstrained length of the actuator's braided sleeve in meters. [Unit: m]
- *Initial Diameter* (`d0`): This represents the initial diameter of the actuator in meters. [Unit: m]

- *Initial Braid Angle* (`theta0`): The angle between the yarn direction and the longitudinal axis of the actuator in its undeformed state, expressed in degrees. [Unit: deg]
- *Gauge Pressure* (`gpress`): The internal gauge pressure of the actuator, which influences the yarn extensibility. [Unit: kPa]

The volume of the actuator, denoted by `volume`, is calculated using a series of steps that account for the actuator's deformation and the geometric relationships of its components. The steps involved in this computation are outlined below.

1. *Yarn Length Calculation*
   The initial yarn length, denoted as `yarn_length`, is computed based on the initial length of the actuator and the initial braid angle. This calculation assumes that the yarn's length remains constant under initial conditions:

$$\texttt{yarn\_length} = \frac{l_0}{\cos(\theta_0)} \tag{14}$$

   where:

   - l_0 is the initial length of the actuator's braided sleeve.
   - θ_0 is the initial braid angle of the actuator.

2. *Number of Turns Calculation*
   The number of turns around the braid, denoted as `num_turns`, is determined using the initial yarn length, the initial diameter, and the initial braid angle:

$$\texttt{num\_turns} = \frac{\sin(\theta_0) \cdot \texttt{yarn\_length}}{\pi d_0} \tag{15}$$

   where:

   - d_0 is the initial diameter of the actuator.

3. *Current Length Calculation*
   The current length of the actuator, denoted as l_1, after deformation is calculated by considering the initial length and the applied strain:

$$l_1 = l_0 \left(1 - \frac{\text{strain}}{100}\right) \tag{16}$$

   where:

- `strain` is the longitudinal contractile strain expressed as a percentage.

4. *Current Braid Angle Calculation*
   The current braid angle, denoted as θ_1, is recalculated considering the geometric relationship between the initial and current lengths:

$$\theta_1 = \cos^{-1}\left(\frac{l_1 \cos(\theta_0)}{l_0}\right) \tag{17}$$

   This equation captures the dependency of the braid angle on the contraction of the actuator.

5. *Current Diameter Calculation*
   The current diameter of the actuator, denoted as d_1, is determined by first calculating the current perimeter of the braid, `curr_perimeter`, using the current yarn length and the current braid angle:

$$\texttt{curr\_perimeter} = \sin(\theta_1) \cdot \texttt{yarn\_length} \tag{18}$$

   The current diameter d_1 is then calculated as:

$$d_1 = \frac{\texttt{curr\_perimeter}}{\texttt{num\_turns} \cdot \pi} \tag{19}$$

6. *Volume Calculation*
   Finally, the volume of the actuator, `volume`, is determined by multiplying the cross-sectional area of the actuator with its current length:

$$\texttt{volume} = \pi \frac{d_1^2}{4} \cdot l_1 \tag{20}$$

   This calculated volume is crucial for modeling the actuator's behavior, as it directly influences the internal pressure and mechanical response under different loading conditions.

The function `vol_gen` provides a comprehensive approach to calculating the volume of a McKibben actuator, accounting for the effects of strain, braid angle, and yarn rigidity. Understanding these geometric transformations allows for accurate modeling of the actuator's physical properties and its response to external stimuli, which is essential for optimizing performance in various applications.

The function `vol_gen` is called as follows:

```
[init_vol,~,~] = vol_gen(0,param.k,param.l0,param.d0,param.theta0,0);
```

It is defined as follows:

```
function[volume,yarn_length,d1]=vol_gen(strain,k,l0,d0,theta0,gpress)
  yarn_length = l0 / cosd(theta0);
  init_perimeter = sind(theta0) * yarn_length;
  num_turns = init_perimeter / (pi*d0);
  l1 = l0 * (1-(strain/100));
  theta1 = acosd((l1*cosd(theta0))/l0);
  yarn_length = yarn_length * (1+k*gpress);
  curr_perimeter = sind(theta1) * yarn_length;
  d1 = (curr_perimeter / num_turns) / pi;
  volume = pi * (d1^2)/4 * l1;
end
```

Initialization of Variables for Iteration

The initialization of variables is a critical step in the iterative process used to simulate the behavior of the McKibben actuator under various conditions. Proper initialization ensures that the model enters the loop correctly and that all parameters are set to reasonable starting values. This section outlines the initial values assigned to variables used in the iteration process, particularly for simulating thermal and volumetric changes within the actuator.

The initialized variables are described as follows:

- *Current Iteration Saturation Pressure* (`curr_iter_psat`): This variable is initialized to the atmospheric pressure `param.patm`, a value high enough to ensure entry into the loop during the first iteration. [Unit: kPa]
- *Current Iteration Saturation Temperature* (`curr_iter_tsat`): This variable is initialized to the boiling point `param.bp`, ensuring that the model starts with a high enough temperature to enter the iteration loop. [Unit: °C]
- *Current Iteration Liquid Heat Transfer Rate* (`curr_iter_Qliq`): Initialized to 0, this variable tracks the heat transfer rate in the liquid phase during each iteration. [Unit: W]
- *Current Iteration Vapor Heat Transfer Rate* (`curr_iter_Qvap`): Initialized to 0, this variable accounts for the heat transfer rate in the vapor phase during each iteration. [Unit: W]
- *Current Iteration Coil Heat Transfer Rate* (`curr_iter_Qcoil`): Initialized to 0, this variable represents the heat transfer rate from the coil during each iteration. [Unit: W]

- *Current Iteration Volume* (`curr_iter_vol`): This variable is initialized to the actuator's initial volume `init_vol` calculated previously, representing the starting volume at the beginning of the iteration process. [Unit: m^3]
- *Last Iteration Volume* (`last_iter_vol`): Initialized to 0, this variable stores the volume from the previous iteration, allowing for comparison and convergence checking. [Unit: m^3]
- *Last Iteration Saturation Pressure* (`last_iter_psat`): Initialized to 0, this variable stores the saturation pressure from the previous iteration, aiding in the convergence analysis of the pressure values. [Unit: kPa]
- *Last Iteration Saturation Temperature* (`last_iter_tsat`): Initialized to 0, this variable keeps track of the saturation temperature from the last iteration, ensuring proper monitoring of temperature convergence. [Unit: °C]

The initialization of these variables is implemented in MATLAB as follows:

```
curr_iter_psat = param.patm;
curr_iter_tsat = param.bp;
curr_iter_Qliq = 0;
curr_iter_Qvap = 0;
curr_iter_Qcoil = 0;
curr_iter_vol = init_vol;
last_iter_vol = 0;
last_iter_psat = 0;
last_iter_tsat = 0;
```

This initialization ensures that the simulation begins with appropriate values, allowing the iterative process to properly evaluate changes in pressure, temperature, and heat transfer within the system.

Heat Generation Due to Electrical Power Input and Heat Dissipation

The heat generation in the actuator system is influenced by both electrical power input and heat dissipation processes. The heat transfer rate, denoted as q_{heat}, is modeled using the coil's surface area, heat transfer coefficient, and the temperature difference between two states. This calculation is crucial for understanding the thermal dynamics of the actuator during operation.

The heat generation is calculated as follows:

$$q_{\text{heat}} = \text{area\_coil} \cdot h_3 \cdot (\text{state}(4) - \text{state}(7)) \tag{21}$$

- *Coil Surface Area* (`area_coil`): This represents the surface area of the coil, which is involved in heat transfer. [Unit: m^2]

- *Heat Transfer Coefficient* (`h3`): This coefficient reflects the efficiency of heat transfer between the actuator and its surroundings. [Unit: $\mathrm{W}/(\mathrm{m}^2 \cdot \mathrm{K})$]
- *Temperature Difference* (`state(4) - state(7)`): This term represents the temperature difference between two specific states within the system, driving the heat transfer. [Unit: K]

Important Note: The heat transfer expression used in this model has the potential to violate the second law of thermodynamics, depending on system constraints and assumptions. This aspect should be carefully considered during analysis to ensure realistic and physically consistent results.
The modeling of heat generation through this equation plays a critical role in assessing the thermal behavior of the McKibben actuator, particularly in applications where heat management is crucial for maintaining performance and preventing overheating.

Heat Dissipation

The following MATLAB code illustrates the computation of electrical power input and heat dissipation:

```
q_elect = power;
q_heat = heatDissipation(param.area_coil,param.h3,state(4),state(2));
```

The function `heatDissipation` is called within the heat generation model to calculate the energy dissipation from the coil to the surrounding medium. The function takes in several parameters related to the physical properties of the system.
The heat dissipation equation can be written as:

$$q_{\text{heat}} = A \cdot h \cdot (T - T_{\text{amb}}) \tag{22}$$

where:

- $A = \texttt{area\_coil} = 5.4978 \times 10^{-4}\ \mathrm{m}^2$
  The surface area of the heating coil.
- $h = \texttt{h3} = 1500\ \mathrm{W}/(\mathrm{m}^2\mathrm{K})$
  The heat transfer coefficient between the metal and the surrounding liquid.
- $T = \texttt{state(4)} = \text{param.bp} + \frac{\text{state(7)}}{\text{param.mass\_coil} \cdot \text{param.cp\_coil}}$
  The temperature of the object (in K), calculated from the system state and physical properties, where param.bp is the boiling point and state(7) relates to the coil's internal energy.
- $T_{\text{amb}} = \texttt{state(2)} = \text{curr\_iter\_tsat} = \text{param.bp} = 100 + 273.15\ \mathrm{K}$
  This is the ambient temperature, initialized to the boiling point at atmospheric pressure.

In the first iteration, `curr_iter_tsat` is set to a high value (equal to the boiling point) to ensure that the loop is entered.

```
function q = heatDissipation(A, h, T, Tamb)
  q = A * h * (T - Tamb);
end
```

The internal energy is updated based on the heat exchange in the system. The change in heat of the coil is given by:

$$\Delta Q_{\text{coil}} = \text{param.simstep} \cdot (q_{\text{elect}} - q_{\text{heat}}) \tag{23}$$

The current internal energy of the coil is updated as:

$$\text{curr\_iter\_Qcoil} = \text{state}(7) + \Delta Q_{\text{coil}} \tag{24}$$

where state(7) is initialized as curr_iter_Qcoil.
In the steam generation process, heat flux components are modeled as follows:
The heat flux from the coil to the fluid is:

$$q_1 = q_{\text{heat}} \tag{25}$$

The heat dissipation from the gas to the liquid is:

$$q_2 = \text{cross\_area} \cdot h2 \cdot (\text{state}(2) - \text{state}(3)) = \text{cross\_area} \cdot h2 \cdot (\text{CurrIter}_{\text{tsat}} - \text{state}(3)) \tag{26}$$

The heat dissipation from the liquid to the ambient is:

$$q_3 = \text{surfarea} \cdot h1 \cdot (\text{state}(3) - 373.15) \tag{27}$$

The heat flux into the gas is given by the difference between the heat transferred from the coil to the fluid and the heat dissipation from the gas to the liquid:

$$q_{\text{vap}} = q_1 - q_2 \quad [\text{W}] \tag{28}$$

The heat flux into the liquid is defined as the difference between the heat dissipation from the gas to the liquid and from the liquid to the ambient:

$$q_{\text{liq}} = q_2 - q_3 \quad [\text{W}] \tag{29}$$

The internal energy of the liquid and vapor is updated based on the respective heat fluxes:

$$\Delta Q_{\text{liq}} = \text{param.simstep} \cdot q_{\text{liq}} \tag{30}$$

$$\Delta Q_{\text{vap}} = \text{param.simstep} \cdot q_{\text{vap}} \tag{31}$$

Thus, the current internal energy for the liquid and vapor phases is updated as:

$$\texttt{curr\_iter\_Qliq} = x_5 + \Delta Q_{\text{liq}} \tag{32}$$

$$\texttt{curr\_iter\_Qvap} = x_6 + \Delta Q_{\text{vap}} \tag{33}$$

To ensure that the internal energy of the vapor does not become negative, the updated internal energy is constrained using a saturation function:

$$\texttt{curr\_iter\_Qvap} = \text{saturation}(\text{curr\_iter\_Qvap}, 0, \infty) \tag{34}$$

Saturation function

The `saturation` function is designed to constrain a given value within specified minimum and maximum bounds. This function ensures that the variable does not exceed physical limits or fall below acceptable thresholds, such as preventing negative internal energy values.
The function operates as follows:
If the input value (`val`) is less than the specified minimum bound (`min`), the function returns the minimum value. If the input value exceeds the specified maximum bound (`max`), the function returns the maximum value. If the input value lies within the range defined by the minimum and maximum bounds, the function simply returns the input value unchanged.

```
function out = saturation(val, min, max)
  if val < min
    out = min;
  elseif val > max
    out = max;
  else
    out = val;
  end
end
```

Calculation of Steam Mass

The mass of steam is calculated based on the current internal energy of the vapor phase. The total energy available for vaporization is represented by and the mass of steam is computed using the following relation:

$$m_{vap} = \frac{\texttt{curr\_iter\_Qvap}}{L} \tag{35}$$

where:

- `curr_iter_Qvap` is the total energy available for vaporization [J].

- L is the latent heat of vaporization [J/kg], a material property of the substance.

This equation is derived from the principle that the amount of energy required to convert a mass of liquid into vapor is directly proportional to the latent heat of vaporization. Thus, the mass of steam produced is obtained by dividing the total energy by the latent heat.

Calculation of Current Volume

The current volume of the vessel is computed based on the strain in the system and the pressure from the last iteration's gauge pressure value. The equation used to calculate the volume is:

$$\texttt{curr\_iter\_vol} = \texttt{vol\_gen}(\text{strain}, k, l_0, d_0, \theta_0, P_{\text{gauge}}) \tag{36}$$

where:

- `vol_gen` is a function that computes the volume based on the mechanical strain and system parameters.
- k is the stiffness coefficient.
- l_0, d_0, and θ_0 are the initial length, diameter, and angular parameters, respectively.
- $P_{\text{gauge}} = \text{state}(9) - P_{\text{atm}}$ is the gauge pressure, calculated as the difference between the current pressure in state(9) and atmospheric pressure (P_{atm}).

After computing the volume, the dead volume of the system is added to account for any non-reducible space inside the vessel:

$$\texttt{curr\_iter\_vol} = \texttt{curr\_iter\_vol} + \texttt{param.dead\_vol} \tag{37}$$

Calculation of Saturation Pressure and Temperature

The new values for saturation pressure (P_{sat}) and saturation temperature (T_{sat}) are determined using the mass of steam in the system and the current volume. The relationship used for this calculation is given by the function:

$$\texttt{pressureInsideVesselPoly}(m_{\text{vap}}, \texttt{curr\_iter\_vol}, \texttt{init\_vol}) \tag{38}$$

where:

- m_{vap} is the mass of steam in the vessel [kg].
- `curr_iter_vol` is the current volume of the vessel [m^3].

- `init_vol` is the initial volume of the vessel [m^3].

The function `pressureInsideVesselPoly` calculates the pressure and temperature using the ideal gas law and a polynomial approximation of the saturation curve of water. The saturation temperature is approximated as:

$$T = a \cdot P^b + c \tag{39}$$

where:

- $a = 14.1697$, $b = 0.1990$, and $c = 231.9364$ are empirical constants derived from the saturation curve of water for temperatures between 373.15 K (100°C) and 414.15 K (140°C).
- P is the absolute pressure [Pa].

The steps for calculating the pressure and temperature are as follows:
Calculate the total mass of fluid (m_{total}) in the system using the density of liquid water, $\rho_{\text{l}} = 1000$ kg/m^3, and the initial volume:

$$m_{\text{total}} = \rho_{\text{l}} \cdot \texttt{init\_vol} \tag{40}$$

Convert the mass of steam (m_{steam}) into moles:

$$n = \frac{m_{\text{steam}} \cdot 1000}{M_{\text{water}}} \tag{41}$$

where $M_{\text{water}} = 18.015$ g/mol is the molar mass of water, and the mass is converted from kg to g.
Compute the volume occupied by steam ($\text{vol}_{\text{steam}}$) by subtracting the liquid volume from the total current volume:

$$\texttt{vol\_steam} = \texttt{curr\_vol} - \frac{m\text{total} - m_{\text{steam}}}{\rho_{\text{l}}} \tag{42}$$

Using the ideal gas law, $PV = nRT$, iteratively compute the pressure and temperature. Starting with an initial guess for pressure (p_{curr}), the temperature is calculated using the polynomial approximation, and the pressure is updated using:

$$p_{\text{curr}} = \frac{nRT}{\text{vol}_{\text{steam}}} \tag{43}$$

where $R = 8.314$ J/(mol · K) is the ideal gas constant. This process is repeated until the change in pressure is below a specified threshold (0.1 Pa).
Finally, the function returns the computed pressure (p_{curr}) and temperature (T), which are then saturated.

```
curr_iter_psat = saturation(curr_iter_psat, param.patm, inf);
curr_iter_tsat = saturation(curr_iter_tsat, param.bp, inf);
```

State Variables and Equations

The system under consideration is described by the following state variables:

1. *Temperature Difference Calculation*:

$$x_1 = \frac{t_{sat} - x_2}{0.1} \tag{44}$$

   where x_1 represents the rate of change of temperature, with x_2 initialized to the saturation temperature t_{sat}, and 0.1 being the simulation step size.

2. *Saturation Temperature*:

$$x_2 = t_{sat} \tag{45}$$

   The saturation temperature t_{sat} is constrained within the range $[bp, \infty]$, where bp is the boiling point.

3. *Liquid Temperature*:

$$x_3 = bp + \frac{x_5}{\text{mass} \times c_{p,liq}} \tag{46}$$

   where:

   - $bp = 373.15\,\text{K}$ (boiling point of water)
   - $\text{mass} = 0.02\,\text{kg}$ (fluid mass inside the system)
   - $c_{p,liq} = 4186\,\text{J/kg} \cdot \text{K}$ (heat capacity of liquid water)

4. *Coil Temperature*:

$$x_4 = bp + \frac{x_7}{\text{mass}_{coil} \times c_{p,coil}} \tag{47}$$

   where:

   - $\text{mass}_{coil} = 0.003\,\text{kg}$ (mass of the heating coil)
   - $c_{p,coil} = 500\,\text{J/kg} \cdot \text{K}$ (heat capacity of the heating coil)

5. *Heat Added to the Liquid*:

$$x_5 = Q_{liq} \tag{48}$$

   The heat added to the liquid evolves according to:

$$Q_{liq} = x_5 + dQ_{liq} = x_5 + 0.1 \cdot q_{liq} \tag{49}$$

   where Q_{liq} is initialized to 0.

6. *Heat Added to the Vapor*:
$$x_6 = Q_{vap} \tag{50}$$
The heat added to the vapor evolves according to:
$$Q_{vap} = x_6 + dQ_{vap} = x_6 + 0.1 \cdot q_{vap} \tag{51}$$
where Q_{vap} is initialized to 0.

7. *Heat Added to the Coil*:
$$x_7 = Q_{coil} \tag{52}$$
The heat added to the coil is initialized at 0.

8. *Diameter of the System*:
$$x_8 = d_1 \tag{53}$$
where d_1 is the diameter.

9. *Saturation Pressure*:
$$x_9 = p_{sat} \tag{54}$$
The saturation pressure is constrained by the range $[p_{atm}, \infty]$.

10. *Boiling Point Indicator*:
$$x_{10} = (x_3 \geq bp) \times 1 \tag{55}$$
This variable acts as an indicator, which is set to 1 if the liquid temperature x_3 reaches or exceeds the boiling point bp. The result of this comparison is a Boolean value: true (1) if x_3 is greater than or equal to param.bp, and false (0) if x_3 is less than param.bp. The multiplication of the Boolean result by 1 serves to explicitly convert the Boolean value into a numeric format.

Output Calculations

The *gauge pressure* is calculated by subtracting the atmospheric pressure from the current saturation pressure:
$$g_{press} = \texttt{curr\_iter\_psat} - p_{atm} \tag{56}$$
where:

- `gpress` is the gauge pressure in Pascals [Pa].
- `curr_iter_psat` is the current iteration value of the saturation pressure.
- p_{atm} is the atmospheric pressure, as defined by the system parameters.

The *output force* generated by the system is calculated using a force generation function based on the gauge pressure and strain values. The tension is linearly proportional to the pressure and depends on the braid angle, which varies between $0° < \theta < 90°$. The generated force is a monotonic function of the braid angle, and is determined using the geometric parameters of the actuator.
The force is calculated using the following equation:

$$F = \frac{\pi D_0^2 P'}{4} \left(3\cos^2\theta - 1\right) \tag{57}$$

where:

- F is the force generated [N].
- D_0 is the initial diameter of the actuator.
- P' is the internal gauge pressure [Pa].
- θ is the braid angle, which is a function of the strain applied to the actuator.

Equation (57) is derived from the geometric analysis of the braided actuator as described by Chou et al. [13].

Force Generation Function

The force generation is implemented in the function `force_gen`, which uses the following steps:

- *Calculate Yarn Length*: The yarn length (L_{yarn}) remains constant and is determined by the initial length of the actuators's braided sleevel_0 and the initial braid angle θ_0:

$$L_{yarn} = \frac{l_0}{\cos\theta_0} \tag{58}$$

- *Calculate Number of Turns*: The initial perimeter of the yarn and the number of turns are determined using the initial braid angle and initial diameter:

$$\texttt{init\_perimeter} = L_{yarn} \cdot \sin\theta_0 \tag{59}$$

$$\texttt{num\_turns} = \frac{\texttt{init\_perimeter}}{\pi D_0} \tag{60}$$

- *Calculate Current Length and New Braid Angle*: The new length (l_1) is calculated based on the applied strain, and the new braid angle (θ_1) is determined as:

$$l_1 = l_0 \left(1 - \frac{\texttt{strain}}{100}\right) \tag{61}$$

$$\theta_1 = \cos^{-1}\left(\frac{l_1 \cos\theta_0}{l_0}\right) \tag{62}$$

- *Calculate Force*: The final force output is computed using:

$$F = \frac{\texttt{gpress} \times L_{yarn}^2}{4\pi \times \text{num\_turns}^2} \left(3\cos^2\theta_1 - 1\right) \tag{63}$$

Conclusion

The proposed model for the soft actuator is derived from the fundamental physics of the actuator's components and their general behavior. However it is important to note that this model is not an exact representation of reality: since each actuator is handcrafted, variations inevitably arise between individual actuators. This inconsistency introduces challenges in terms of reliability and accuracy of the results obtained from the model.

An alternative modeling approach considered was to develop a data-driven model, but this approach faced limitations due to the novelty of these soft actuators, which were first developed by D. Fonseca in 2021. Since then, the design and characteristics of the actuators have undergone numerous modifications. During my time on the project, actuator availability was limited—only three actuators were accessible—due to the ongoing advancements in D. Fonseca's research and the significant time required for the production of each actuator.

Consequently, the unreliability of the actuators compared to the model had to be factored into the experimentation process. Handcrafting introduces variations, such as differences in coil length, water content, or the thickness of the silicone tubing, all of which affect the actuator's response to the same power input. These discrepancies contribute to a degree of error and hinder the creation of a precise, universally applicable model.

3.3 Control Methods

Based on the system model, design a suitable controller to achieve the desired performance criteria, such as position control, force control, or trajectory tracking. Common control techniques for such systems include PID control, state feedback control, or model predictive control (MPC)

3.3.1 ON-OFF controller

The On-Off controller, also known as a bang-bang controller, is one of the simplest control strategies, widely used in applications where precise adjustment of the system is not strictly necessary, but maintaining the process variable within an acceptable range is essential. This type of controller operates by switching the control output between two discrete states: "on" and "off," depending on whether the measured process variable is above or below a set-point. The straightforward nature of the On-Off controller makes it an effective solution for systems requiring robust and uncomplicated control mechanisms.

Working Principle

The On-Off controller works by continuously comparing the measured process variable (e.g., pressure) to a predefined set-point and based on this comparison, the controller generates a control signal that either activates ("on") or deactivates ("off") the actuator. This binary action provides a simple means of driving the process variable towards the desired value, with switching occurring directly based on the set-point:

- **On State**: When the measured process variable is below the set-point, the controller activates the actuator to drive the process variable to increase towards the set-point.
- **Off State**: When the measured process variable exceeds the set-point, the controller deactivates the actuator, allowing the process variable to decrease or stabilize.

Implementation of On-Off Control

The On-Off controller was designed to regulate the system's behaviour within a desired range of pressures, specifically targeting operational stability. The control logic can be mathematically represented as follows:

$$u(t) = \begin{cases} 1, & \text{if } P_{\text{measured}} < P_{\text{setpoint}}, \\ 0, & \text{if } P_{\text{measured}} > P_{\text{setpoint}}. \end{cases}$$

where:

- $u(t)$: Control output at time t (1 = "on", 0 = "off").

- P_{measured}: The current measured pressure.
- P_{setpoint}: The target pressure setpoint.

MATLAB Code Implementation

The following MATLAB code demonstrates the implementation of the On-Off controller:

```
% Define setpoint
setpoint = 25; % Example setpoint in kPa

% Measure current pressure (ref is the measured reference pressure)
if ref < setpoint
  power_supp = 1; % Power supply on
elseif ref > setpoint
  power_supp = 0; % Power supply off
end
```

Operational Characteristics

The key characteristics of the On-Off controller include:

- **Switching Behavior**: The controller turns the power supply connected to the actuator on when the measured pressure falls below the set-point and turns it off when it exceeds the set-point and since no hysteresis is present, the system switches immediately as soon as the set-point is crossed.
- **Response Time**: The On-Off controller provides an immediate response to deviations from the set-point, making it suitable for applications that require prompt corrective actions.
- **Simplicity and Robustness**: Due to its simple design, the On-Off controller requires minimal computational resources and is inherently robust against sudden changes or disturbances in the system: this simplicity makes it a reliable option in basic control scenarios.

The On-Off controller offers a straightforward yet effective approach for maintaining process variables within acceptable limits and its binary action provides a robust control mechanism that is easy to implement and maintain. This form of control is valuable in applications where complex PID or adaptive controllers may be overly sophisticated or unnecessary, and where maintaining operation within a specific range is the primary objective. The downside is the oscillating range it maintains not allowing a stable output as well as other controllers.

3.3.2 PI Controller: Experimental and Simulation Tuning

The PI (Proportional-Integral) controller is a control strategy that combines proportional and integral actions to reduce steady-state error and ensure smoother transitions to the set-point in dynamic systems. By continuously adjusting the control output based on both current error and accumulated error over time, the PI controller enhances the performance of simpler control methods, such as On-Off controllers, making it suitable for more precise applications.

Working Principle

The PI controller works by computing the error between the desired set-point and the measured process variable. It then applies two correction actions:

- **Proportional Action (K_p):** Responds to the current error magnitude. The larger the error, the greater the corrective effort, helping to bring the process variable closer to the set-point.
- **Integral Action (K_i):** Addresses accumulated error over time, ensuring that even small, persistent errors are eliminated. This is particularly useful in eliminating steady-state errors.

The control signal output from the PI controller is given by:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau$$

Where:

- $u(t)$: Control signal at time t (e.g., PWM duty cycle).
- $e(t)$: Error at time t, i.e., $P_{\text{setpoint}} - P_{\text{measured}}$.
- K_p: Proportional gain, which adjusts the response to the current error.
- K_i: Integral gain, which adjusts the response to the accumulated error.

Experimental Tuning of the PI Controller

The initial tuning of the PI controller was performed using experimental data collected from a latex-based actuator. The values of K_p and K_i were derived through hands-on control using a programmable power supply. Voltage and pressure data were gathered to analyze the actuator's behavior, which guided the initial tuning of the gains.
Subsequently, the collected data was processed using MATLAB's System Identification Toolbox to model the system dynamics. The PID Tuner in MATLAB was then employed to

further refine K_p and K_i, ensuring a well-tuned response for accurate pressure regulation. A saturation limit of 255 was applied to the control signal to prevent integral windup, ensuring stable operation of the actuator.
The optimal gains derived from this process were:

- $K_p = 0.1$
- $K_i = 0.02$

Simulation-Based Tuning of the PI Controller

In addition to experimental tuning, a simulation-based approach was used to further refine the PI controller. This method involved systematically adjusting K_p and K_i to achieve optimal performance across a range of pressure references, using simulations to iteratively improve the controller's response.
The tuning process was guided by the following steps:

1. **Initial Setup:** Starting with low values of K_p and K_i to establish a stable baseline.
2. **Adjust K_p:** Increasing K_p gradually to reduce steady-state errors, while carefully monitoring for oscillations.
3. **Balance K_p and K_i:** Adjusting K_i to complement K_p, eliminating steady-state errors and fine-tuning system stability.
4. **Monitor System Response:** Observing the system's behavior to avoid issues such as overshoot, oscillations, or excessive settling time.

The best tuning values from the simulation were:

- $K_p = 0.5$
- $K_i = 0.0137$

These values provided satisfactory performance at higher pressure references, but exhibited some delay at lower pressures due to the pre-heating phase of the actuator incorporated into the model.

Operational Characteristics

The PI controller's performance characteristics were assessed through both experimental and simulation data:

- **Error Correction:** The proportional term addresses current errors, while the integral term eliminates steady-state errors.

- **Smooth Response:** The PI controller provides smoother control, minimizing oscillations compared to the On-Off controller.
- **Prevention of Windup:** A saturation limit is applied to prevent integral windup, ensuring stable operation.
- **Adaptability:** The gains derived from both experimental and simulation methods ensure the controller can adapt to varying actuator sizes and materials.

The combination of both experimental and simulation tuning approaches offers flexibility in the controller's design, allowing it to perform well under different operating conditions and pressures.

3.3.3 Adaptive PID Controller

The performance of the PI controller was found to be satisfactory at higher reference pressures but inadequate at lower pressures. To address this, the implementation of a PID controller was considered by adding a derivative term to improve its behavior at low pressures. Furthermore, to enhance the original PI controller, it was decided to make it adaptive, allowing it to adjust its responsiveness based on the reference pressure.
To improve the behavior of the controller at low reference values, a decision was made to incorporate a PID controller that operates exclusively below 25 kPa. Additionally, to enhance the performance of the PI controller at higher pressures, instead of using constant values for K_p and K_i, the behavior of the actuator was analyzed using real data to understand its actual response.

PI Controller

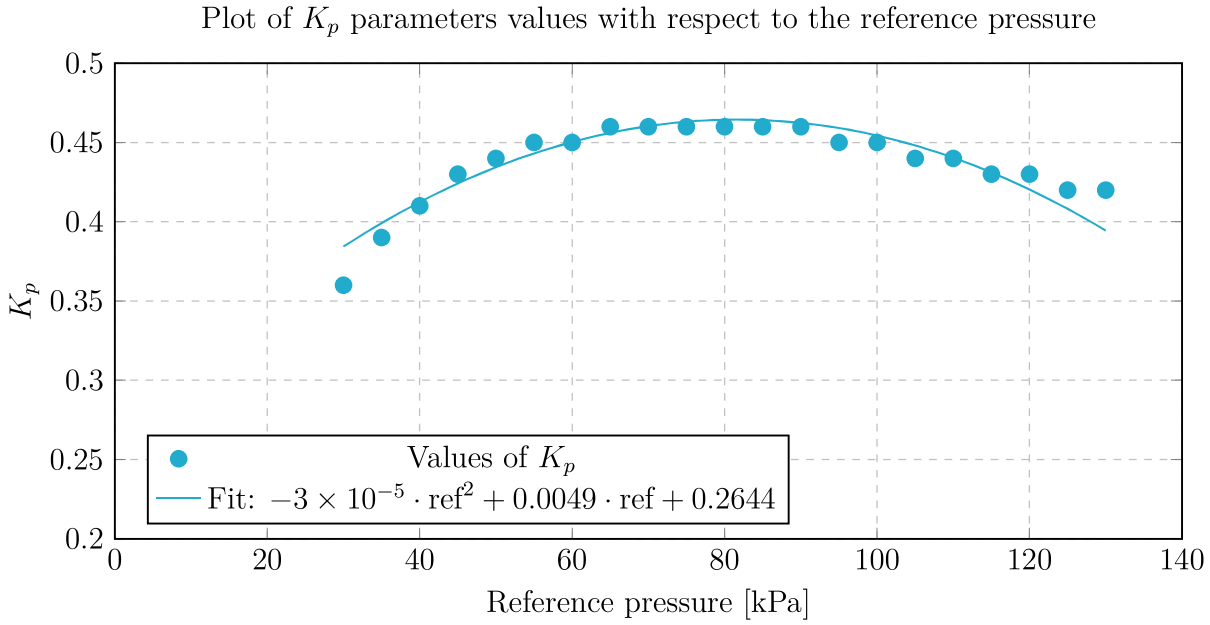
The PI controller operates when the reference pressure is greater than 25 kPa. For reference pressures between 26 kPa and 130 kPa (maximum), the parameters are given by:

$$P_{\text{PI}} = -3 \times 10^{-5} \cdot \text{ref}^2 + 0.0047 \cdot \text{ref} + 0.2644$$
$$I_{\text{PI}} = -7 \times 10^{-7} \cdot \text{ref}^2 + 0.0001 \cdot \text{ref} + 0.0085$$

These equations were derived by finding the best values for K_p and K_i for different reference values:

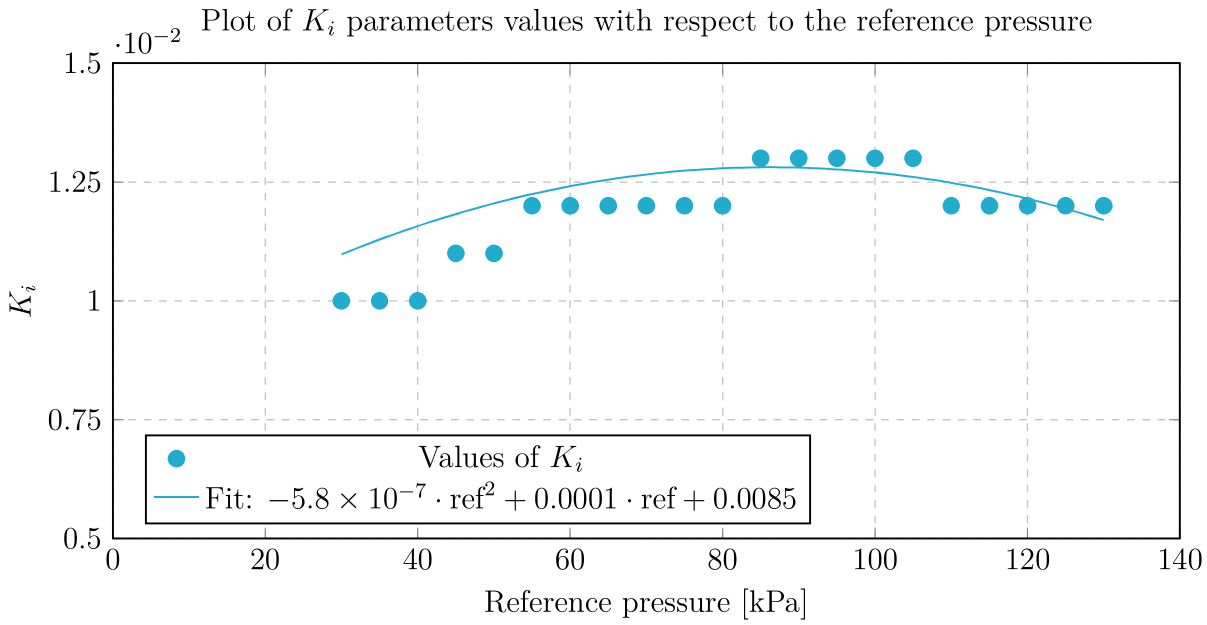
Ref Pressure (kPa)	K_p	K_i
20	0.357	0.00855
17	0.32	0.0075
15	0.29	0.0065
12	0.245	0.005
10	0.2165	0.0038
7	0.102	0.0029
5	0.063	0.0013

Table 1: PID Controller Parameters for Different Reference Pressures



Ref Pressure (kPa)	K_p	K_i
30	0.36	0.01
35	0.39	0.01
40	0.41	0.01
45	0.43	0.011
50	0.44	0.011
55	0.45	0.012
60	0.45	0.012
65	0.46	0.012
70	0.46	0.012
75	0.46	0.012
80	0.46	0.012
85	0.46	0.013
90	0.46	0.013
95	0.45	0.013
100	0.45	0.013
105	0.44	0.013
110	0.44	0.012
115	0.43	0.012
120	0.43	0.012
125	0.42	0.012
130	0.42	0.012

Table 2: PI Controller Parameters for Different Reference Pressures



PID Controller

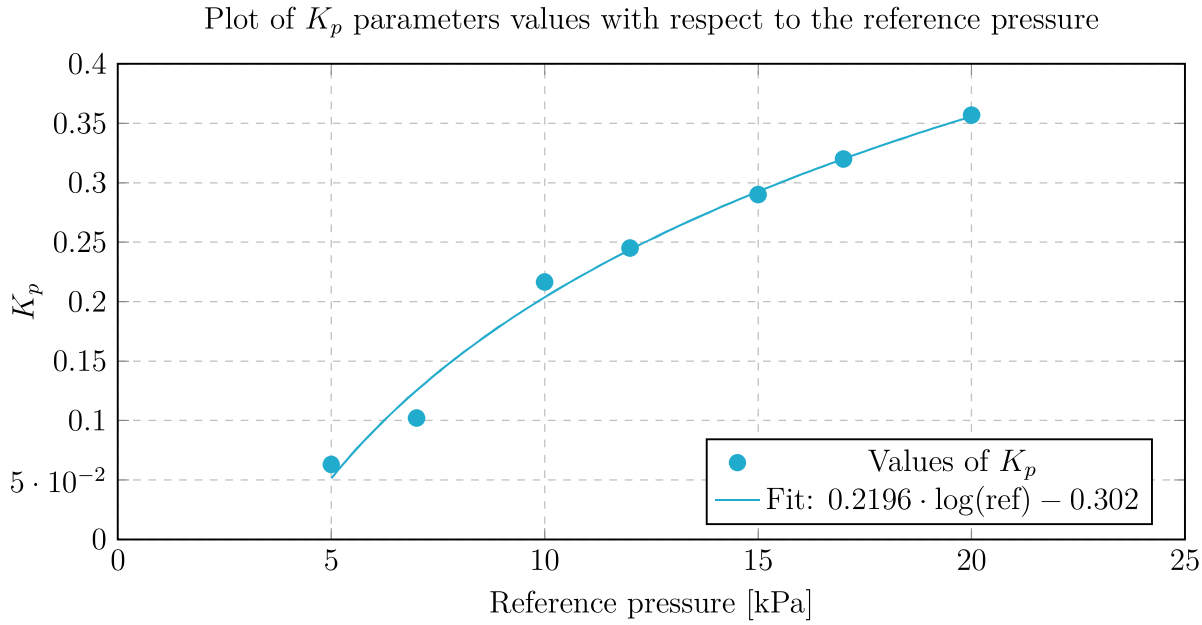
For low pressures (from 25 kPa down), I use a PID controller which allows me to control better at low pressures. The parameters are given by:

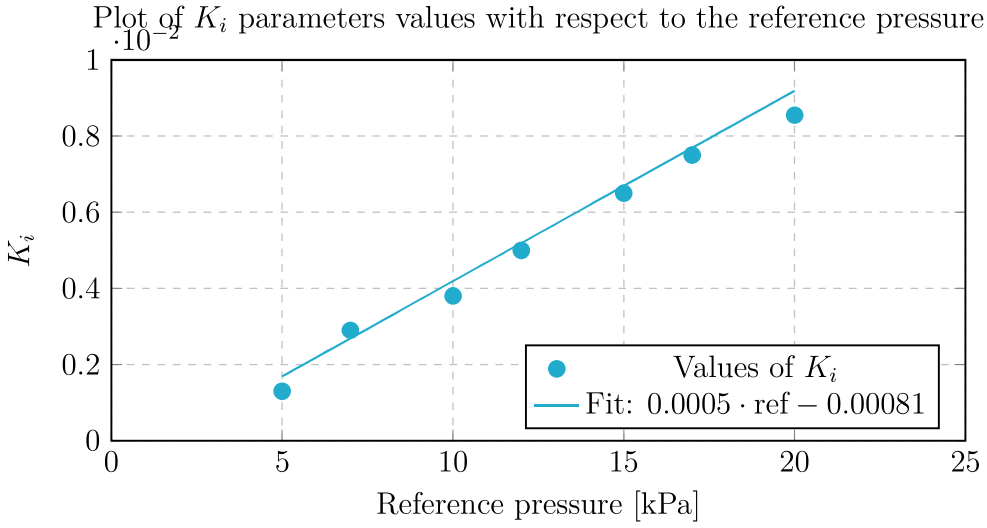
$$P = 0.2196 \cdot \log(\text{ref}) - 0.302$$
$$I = 0.0005 \cdot \text{ref} - 0.00081$$
$$D = 0.15$$

These equations were derived by finding the best values for K_p, K_i, and K_d for different reference values:

Ref Pressure (kPa)	K_p	K_i	K_d
20	0.357	0.00855	0.15
17	0.32	0.0075	0.15
15	0.29	0.0065	0.15
12	0.245	0.005	0.15
10	0.2165	0.0038	0.15
7	0.102	0.0029	0.15
5	0.063	0.0013	0.15

Table 3: PID Controller Parameters for Different Reference Pressures





An adaptive PID controller is an advanced control system that dynamically adjusts its proportional, integral, and derivative gains in real-time to maintain optimal control performance in the presence of changing system dynamics and external disturbances. By utilizing adaptive algorithms, the controller continuously monitors and modifies its parameters, ensuring enhanced stability, accuracy, and robustness compared to traditional fixed-parameter PID controllers. This adaptability makes it suitable for applications requiring precise control in environments with nonlinearities, time-varying characteristics, and uncertainties.

In our case, the adaptiveness applies only to the proportional and integral terms, which are adjusted based on specific equations. The derivative term is modified using a conditional statement based on the reference pressure. The corresponding MATLAB code for this adaptive PID controller is as follows:

```
if ref<25
  P = 0.2196*log(ref)-0.302;
  I = 0.0005*ref - 0.00081;
  D = 0.15;
else
  P = -3*10^(-5)*ref^2 + 0.0047*ref + 0.2644;
  I = -7*10^(-7)*ref^2 + 0.0001*ref + 0.0085;
  D = 0;
end
```

This MATLAB code demonstrates the adaptive PID controller's approach to modifying the proportional (P) and integral (I) gains through specific equations based on the reference pressure (ref). The derivative (D) gain is set to a fixed value for lower reference pressures and is zeroed out for higher reference pressures. This conditional adaptation helps in managing the system's control parameters effectively under varying operational conditions.

4 Simulation and Testing

4.1 Testing Set-Up

A custom test rig was designed and developed to conduct both isometric (constant strain) and isotonic (constant stress) experiments on the soft actuators. The force was measured using a parallel beam load cell (TAL220B, HT Sensor Technology Co.) interfaced with an HX711 amplifier and an STM Nucleo 64 board for data acquisition. Displacement measurements were obtained using a Polhemus Liberty tracking system, providing high-precision tracking of actuator movement. [18]
Both the force and displacement measurement systems were calibrated, achieving measurement accuracies of $\pm 0.02\,\mathrm{N}$ and $\pm 0.5\,\mathrm{mm}$, respectively. Additionally, internal temperature and pressure sensors embedded within the actuators provided real-time data, synchronized with the external measurement systems. All sensor data were collected and monitored using a MATLAB script running on an external computer, which ensured accurate synchronization and reliable data acquisition throughout the experiments.
This integrated setup (Fig. 13)allowed for precise control of experimental conditions and real-time monitoring, facilitating comprehensive analysis of the actuator's performance under both isometric and isotonic testing scenarios.

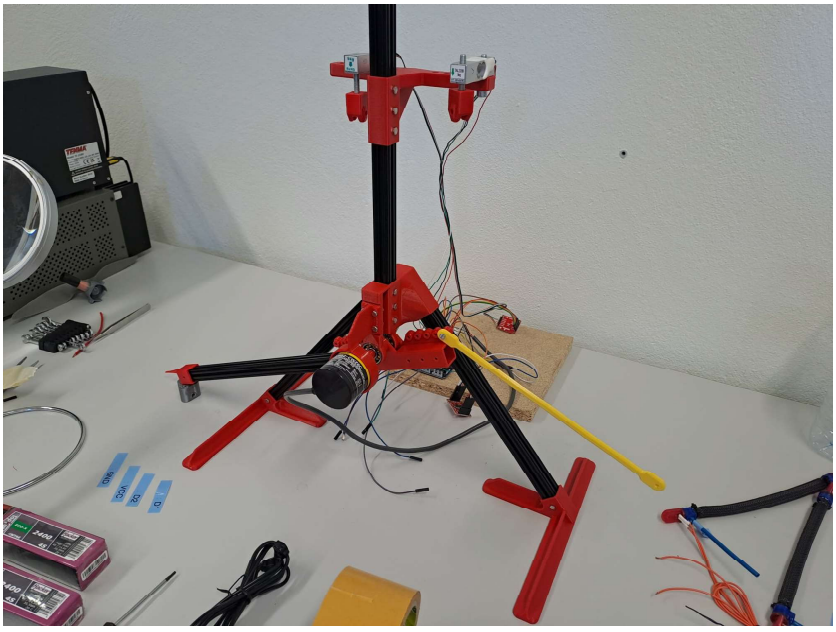


Figure 13: Testing set-up for experiments on the soft actuator

4.1.1 Control System Implementation

This chapter presents the implementation of a control system used for testing a soft actuator. The control system is written in C++ and runs on a microcontroller platform. The code handles sensor inputs, performs control calculations, and manages actuator outputs. The system supports different controllers, including On-Off, PI, and PID controllers, and it measures force, displacement, and pressure to control actuators through Pulse Width Modulation (PWM).

Controller Type Selection

The variable `contr` defines the controller type used in the system. This can be an On-Off controller, a PI controller, or a PID controller. The selection is based on the value of `contr`:

- `0`: On-Off controller
- `1`: PI controller
- `2`: PID controller

```
int contr = 0;
#include "HX711.h"
float maxCurrent = 1.8;
```

The variable `maxCurrent` is defined to limit the maximum current to 1.8A, ensuring the safety of the system components.

Shoulder Control and Load Cell Setup

This section initializes parameters and pin configurations for controlling the shoulder motor and reading data from a load cell. It also sets up actuator parameters and defines variables for storing sensor readings.

```
const byte pwm_6v = 255;
bool ShoulderUp = false;
bool ShoulderDown = false;
const byte UpGatePin = 13;
const byte DownGatePin = 11;
float left_calibration = 462;

#include <TimerOne.h>
float maxPressure = 130;

byte M1 = 4;
byte M2 = 7;
byte E1 = 5;
byte E2 = 6;
```

```
15
16  #define DOUTL 4
17  #define CLKL 5
```

Here, `pwm_6v` is set to control the shoulder motor's PWM value. Boolean flags `ShoulderUp` and `ShoulderDown` track the shoulder's position. Pins `UpGatePin` and `DownGatePin` control the motor's direction. The `force_left` variable stores force readings from the load cell, and `left_calibration` is the calibration factor for the load cell. TimerOne is included to manage timing functions. The `maxPressure` variable defines the maximum pressure the system can apply, and arrays `setCoordinate` and `currCoordinate` store the target and actual actuator positions, respectively. The byte array `msg` is used for data handling. The pins `M1`, `M2`, `E1`, and `E2` control actuators, while `DOUTL` and `CLKL` are for load cell communication.

Main Control Loop

The `loop()` function is the core of the control system's operation. It continuously reads sensor data, computes control signals, and updates actuator commands based on the chosen control strategy. The function consists of the following main tasks:

- Reading Force Data:

```
1     force_left = left_cell.get_units()
```

  Reads the force data from a load cell connected to the actuator and stores it in the variable `force_left`. This data is used for monitoring the force applied by the actuator.

- Reading Set Pressure:

```
1     setCoordinate[0] = (analogRead(A3)/float(1023))*maxPressure
```

  Reads the desired pressure (set pressure) from a potentiometer connected to analog pin `A3`. The raw ADC value (0-1023) is scaled to the actual pressure range using the `maxPressure` variable, which represents the maximum allowable pressure for the actuator. The resulting value is stored in the `setCoordinate[0]` array for actuator `A1`. This section could also handle actuator `A2` by reading from pin `A4`.

- Reading Current Pressure:

```
1     currCoordinate[0] = getPressure(A1)
```

  Reads the current pressure from a sensor connected to analog pin `A1`. The `getPressure()` function converts the raw ADC value to pressure in kilopascals (kPa) and stores it in the `currCoordinate[0]` array for actuator `A1`. Similarly, pressure data for actuator `A2` could be handled using an additional line with pin `A2`.

- Calculating Error:

```
    error[0] = setCoordinate[0] - currCoordinate[0];
```

  Computes the difference (error) between the desired pressure (`setCoordinate[0]`) and the actual measured pressure (`currCoordinate[0]`). This error value is essential for determining the control signal sent to the actuator.

- Adjusting PID Coefficients:

```
    if contr == 2 {
      float kp[2], ki[2], kd[2];
      for (int i = 0; i < 2; i++) {
        if (setCoordinate[i] < 25) {
          kp[i] = 0.2196 * log(setCoordinate[i]) - 0.302;
          ki[i] = 0.0005 * setCoordinate[i] - 0.00081;
          kd[i] = 0.15;
        } else {
          kp[i] = -3e-5 * setCoordinate[i] * setCoordinate[i] +
   0.0047 * setCoordinate[i] + 0.2644;
          ki[i] = -7e-7 * setCoordinate[i] * setCoordinate[i] +
   0.0001 * setCoordinate[i] + 0.0085;
          kd[i] = 0;
        }
      }
    }
```

  Adjusts the PID coefficients based on the set pressure value if the `contr` variable is equal to 2. For low set pressures (`< 25`), logarithmic and linear equations are used to compute the coefficients. For higher pressures, a polynomial equation is applied. This allows the controller to adapt to different operating conditions.

- Calculating Control Signal:

```
    dt = millis() - last_time;
    last_time = last_time + dt;
    if contr == 0 {
      pwm[0] = ONOFFcontrol(error[0], deadband, state);
    } else if contr == 1 {
      pwm[0] = PIcontrol(0, kp_PI, ki_PI, error[0], dt, maxCurrent);
    } else if contr == 2 {
      pwm[0] = PIDcontrol(0, kp[0], ki[0], kd[0], error[0],
   prev_error[0], dt, maxCurrent);
    }
```

  Computes the control signal (`pwm[0]`) based on the selected control strategy (`contr`). The control strategies are:

  - `contr == 0`: ON/OFF control using the `ONOFFcontrol()` function.

- `contr == 1`: PI control using the `PIcontrol()` function with preset coefficients.
- `contr == 2`: PID control using the `PIDcontrol()` function with dynamically adjusted coefficients.

- Sending Control Signal to Actuator:

```
    digitalWrite(M1, LOW);
    analogWrite(E1, pwm[0]);
```

Sends the computed PWM value (`pwm[0]`) to the actuator `A1` through the motor shield. This controls the speed and direction of the actuator.

- Sending Data Over Serial:

```
    Serial.print(int(force_left));
    Serial.print(int(setCoordinate[0]));
    Serial.println(int(currCoordinate[0]));
```

Transmits relevant data, such as force, set pressure, and current pressure, over the serial interface for monitoring or logging purposes.

- Waiting for Confirmation:

```
    waitfornext();
```

Waits for a confirmation signal from the PC before starting the next control loop iteration, ensuring synchronization between the PC and the microcontroller.

Function Definitions
Additional utility functions are defined below:

```
void setPwmFreq(int frequency) {
  int period = (1e6 / frequency);
  Timer1.initialize(period);
  Timer1.stop();
  Timer1.restart();
}
```

The `setPwmFreq()` function sets the frequency for the PWM signal using Timer1. This function allows the frequency to be adjusted dynamically during the system's operation.

```
void handshake() {
  while (!Serial) {;}
  Serial.begin(115200);
  char a = 'b';
  while (a != 'a') {
    a = Serial.read();
  }}
```

The `handshake()` function waits for a connection with the PC and sends an acknowledgment character 'a' until it receives a response, establishing communication between the microcontroller and the PC.

```
void waitfornext() {
  char n = 'a';
  while (n != 'n') {
    n = Serial.read();
  }
}
```

The `waitfornext()` function ensures that the microcontroller waits for a signal from the PC before continuing with the next cycle, synchronizing operations between the microcontroller and the PC.

```
float getPressure(byte pin) {
  int adc;
  float result;
  adc = analogRead(pin);
  result = (adc - 0.1 * 1023) * (60 / (0.8 * 1023));
  result = result * 6.89475729;
  if (result < 0) { result = 0; }
  return result;
}
```

The `getPressure()` function reads data from a Honeywell pressure sensor, converts the ADC value to pressure in kilopascals (kPa), and returns the result. It ensures that negative values are avoided, which might occur due to noise or sensor disconnection.

PI and PID Control Algorithms

The PI and PID control algorithms compute the PWM value based on the proportional, integral, and derivative components of the error. The function `PIcontrol()` is used for PI control, and `PIDcontrol()` is used when the derivative component is also considered.

```
byte PIcontrol(byte idx, float Kp, float Ki, float Error, unsigned
 long dTime, float SaturationLimit) {
  float P, I, LastIntegral, OutputCurrent;
  static float Integral[3];
  LastIntegral = Integral[idx];
  P = Kp * Error;
  Integral[idx] = LastIntegral + (Error * dTime) / 1000;
  I = Ki * Integral[idx];
  if (I < 0) {
    I = 0;
    Integral[idx] = LastIntegral;
  }
  if (P + I > SaturationLimit) {
    if (Error > 0) {
      Integral[idx] = LastIntegral;
```

```
        OutputCurrent = SaturationLimit;
      }
    } else if (P + I < 0) {
      OutputCurrent = 0;
    } else {
      OutputCurrent = P + I;
    }
    return round((OutputCurrent / SaturationLimit) * 255.0);
  }
```

On-Off Controller

The On-Off controller switches the actuator fully on or off based on the error value and a specified deadband. This approach is straightforward but can result in oscillations around the setpoint.

```
byte ONOFFcontrol(float error, float deadband, bool state) {
  byte result;
  float halfband = deadband/2;
  if (error >= halfband) {
    result = 255;
  } else if (error < halfband && error > -halfband && state == true) {
    result = 255;
  } else if (error <= -halfband) {
    result = 0;
  } else if (error < halfband && error > -halfband && state == false)
 {
    result = 0;
  }
  return result;
}
```

This control system forms the core of the experimental setup for testing soft actuators. By selecting appropriate control strategies and parameters, the system can be adapted to a variety of experimental conditions, providing precise data for performance analysis.

4.2 Experimental evaluation of Soft Actuators

In order to understand what and how to compute the various tests the idea was to gather all the informations on the soft actuator beforehand to operate in the best possible conditions.

4.2.1 Isometric shock test

The first aim was to verify that the pressurization rate's limiting factor was the maximum heat dissipation rate, not the Critical Heat Flux (CHF). An isometric shock test was performed by connecting an actuator directly to a programmable DC power supply (TENMA Model 72-2540). A MATLAB control algorithm communicated with the power supply via serial connection, bypassing the Arduino-based system's 25W limit. The actuator was commanded to maintain a relative pressure of 85 kPa, and 110 W of electrical power (3.65 A at 30 V) was applied.

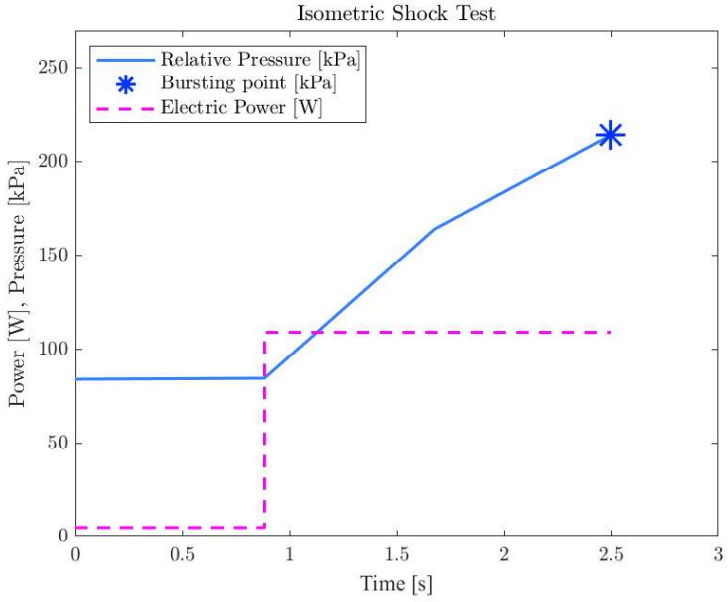


Figure 14: Isometric test results showing measured internal temperature, pressure calculated from temperature, and measured absolute internal pressure over time.

As shown in Figure 14, a pressurization rate of 100 kPa/s was achieved, with the actuator rupturing at 214 kPa, the proposed maximum safe operating pressure is therefore 130 kPa, which is then considered the maximum pressure in each of the subsequent tests.

4.2.2 Temperature sensor test

An isometric test was conducted to assess the feasibility of using internal temperature readings for closed-loop control. A temperature sensor was integrated at the upper-end terminal of a soft actuator, ensuring it was immersed in the gaseous phase (saturated steam) and not in the potentially sub-cooled liquid phase. This setup was intended to have the temperature readings, despite some lag, closely follow the saturation temperature of the fluid corresponding to the internal pressure.

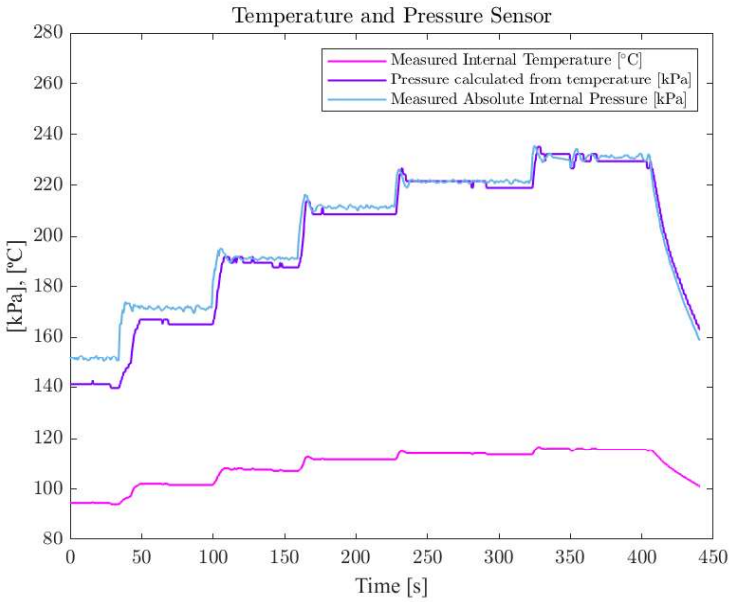


Figure 15: Isometric test results showing measured internal temperature, pressure calculated from temperature, and measured absolute internal pressure over time.

Figure 15 displays the results of the isometric test. The blue curve represents the measured internal temperature in degrees Celsius (°C), while the yellow and orange curves represent the measured absolute internal pressure and the pressure calculated from the temperature, respectively, both in kilopascals (kPa).
From the graph, several key observations can be made:

1. Temperature and Pressure Correlation: The measured internal temperature (blue curve) consistently lags behind the measured absolute internal pressure (yellow curve). This indicates a delay in the temperature sensor's response to changes in pressure.

2. Temperature Readings Lower than Expected: The temperature readings are consistently lower than the calculated saturation temperature corresponding to the mea-

sured internal pressure values. This discrepancy aligns with the explanation that the temperature sensor, while immersed in the gaseous phase, still reports lower temperatures than the theoretical saturation temperature.

3. Pressurization Stages: The graph shows distinct stages of pressurization where the pressure increases in steps. Each step is associated with a corresponding change in the measured internal temperature, although with a noticeable delay and lower values than expected.

4. Saturation Temperature and Pressure: The orange curve, representing the calculated pressure from the temperature, closely follows the pattern of the yellow curve (measured absolute internal pressure), but consistently at a lower value. This reinforces the observation that the internal temperature sensor readings are not perfectly tracking the saturation temperature.

5. Stability at Set Points: The system shows stability at several set pressure points before moving to the next stage, indicating controlled pressurization. The eventual drop in pressure towards the end suggests a release or failure in maintaining the pressurization.

Overall, the graph supports the conclusion that while the internal temperature sensor provides valuable data, its readings show a significant lag and lower values compared to the calculated saturation temperatures based on the measured internal pressures. This behavior must be accounted for in any closed-loop control system using these temperature readings.
This is why, in the experiments, we opted to use a pressure sensor, as it provides more accurate data evaluations.

4.2.3 Response to Variable Power Inputs

To assess the dynamic response of the thermo-electric soft actuator under varying power inputs, a series of experiments were conducted. The primary objective was to quantify the actuator's response time as a function of applied electrical power. The experimental results demonstrated a clear inverse relationship between the power input and the response time, confirming that higher power inputs significantly reduce the time required for the actuator to reach the desired pressure.
As illustrated in Figure 16, the pressure generated within the soft actuator exhibits a positive correlation with the power input. This observation is consistent with theoretical expectations: an increase in electrical power results in enhanced heating of the working fluid. Consequently, the elevated thermal energy facilitates a more rapid phase transition from liquid to gas, thereby increasing the internal pressure of the actuator.

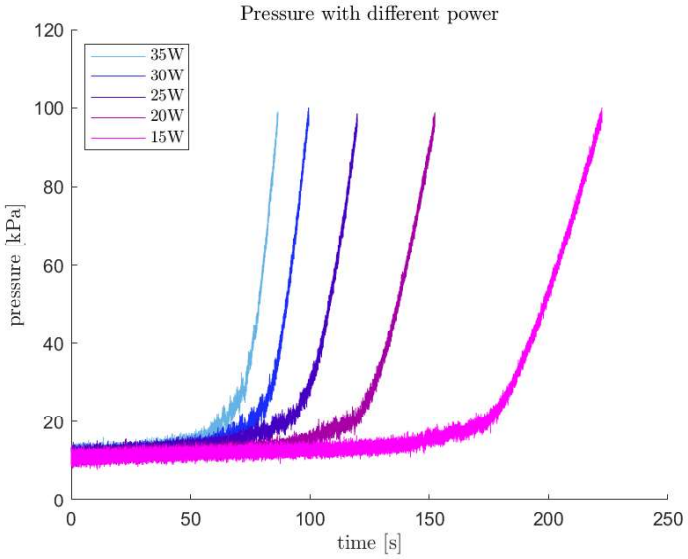


Figure 16: Pressure response of the actuator under different power inputs

These findings underscore the efficacy of power modulation in controlling the actuator's performance and highlight the importance of optimizing power input for desired actuation characteristics.

4.2.4 Cooling-Off Phase

A cooling-off experiment was conducted to examine the thermal dissipation behavior of the soft actuator under different initial reference pressures. In this experiment, the actuator was first pressurized to a set reference pressure, once the desired pressure was achieved, the power supply was turned off, allowing the actuator to cool naturally through convection with ambient air surrounding the external surface of the tube.
Theoretically, the cooling curves for each test are expected to follow a similar exponential decay, governed by Newton's law of cooling. However, as seen in Figure 17, deviations from the theoretical model are evident across trials with different initial reference pressures. This divergence underscores the real-world complexities of soft actuator behavior, where uncontrolled external variables introduce inconsistencies in performance.
These variations can be attributed to several factors:

- Environmental Interference: Factors such as ambient temperature fluctuations, airflow and material properties of the actuator might have contributed to irregular cooling rates.

- Thermal Inertia of the Actuator: The internal material composition and the actuator's structure likely introduced varying thermal resistances, affecting how quickly or slowly the actuator released heat.
- Pressure-Dependent Heat Transfer: The cooling rate might be affected by the initial pressure level, which could alter the internal distribution of temperature and the actuator's thermal conductivity.

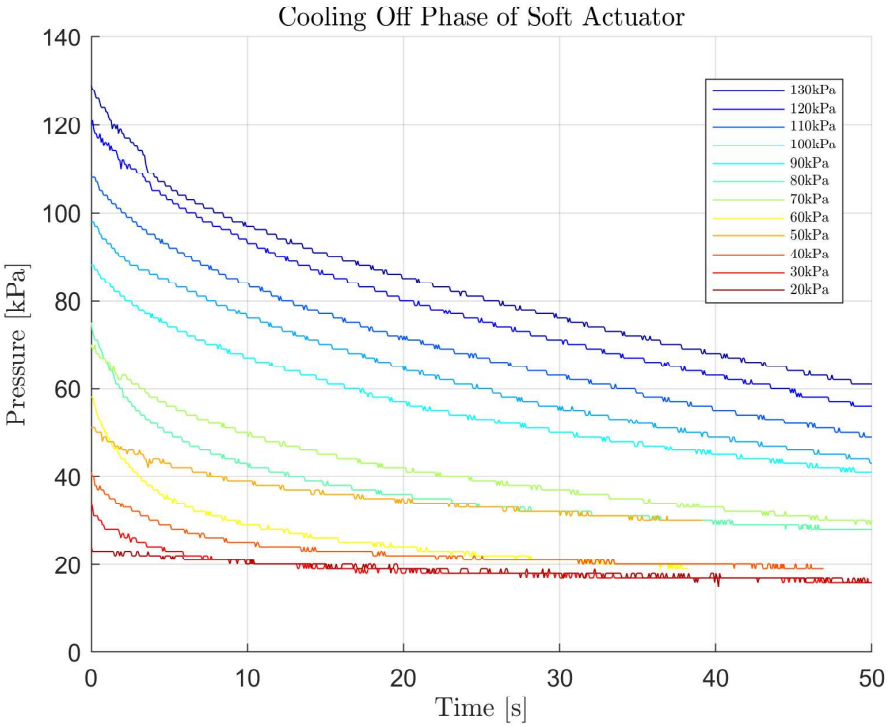


Figure 17: Pressure response of the actuator during the cooling phase with no power input, starting from different initial reference pressures.

Figure 17 illustrates the pressure decay curves observed during the cooling process. Each line corresponds to a test with a different initial reference pressure, showing a noticeable spread in the cooling response.

The results of this experiment suggest that while theoretical models offer valuable insight into expected performance, real-world testing introduces significant uncertainties. The soft actuator exhibits non-ideal thermal behavior, influenced by both external conditions and internal material properties. This variability must be accounted for when designing controllers intended for consistent, reliable operation in uncontrolled environments.

4.2.5 Isotonic Tests

The isotonic tests were performed to obtain the relationship between the angle of displacement and the applied pressure. As shown, the angle, and consequently the displacement, is not as reliable as feedback as the pressure or the force.

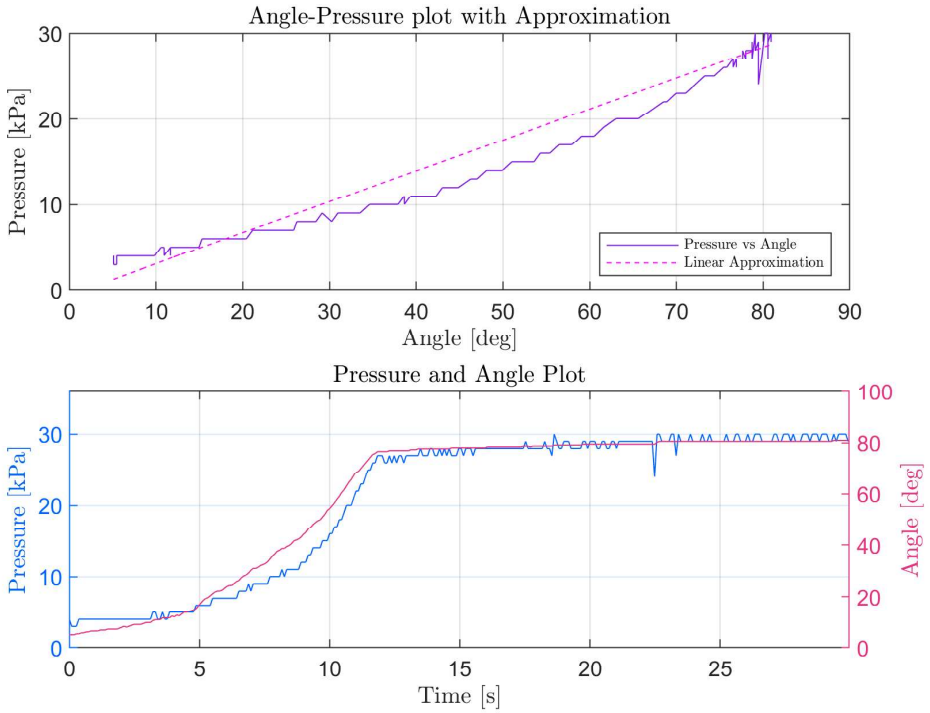


Figure 18: Isotonic test measuring the angle and pressure.

- Linear Strain and Strain Rate Calculation

  The following section outlines the procedure used to calculate both the linear strain and strain rate from angular displacement data captured by an encoder. The encoder is calibrated such that $\theta = 0°$ corresponds to the horizontal configuration of the system. Key geometric parameters used in this calculation include the distance between the top actuator support and the center of the bench (r_s), the vertical distance between the top and bottom actuator supports (s_o), and the distance from the bottom actuator support to the axis of rotation (p_o). The calculations are based on simple geometric relationships between these distances and the angular position of the actuator.

- Geometric Setup

  The geometry of the system is used to relate the angular displacement of the actuator, θ, to the change in the system's length. This is done using trigonometric relationships, as outlined below.

- Angular Calculations

  The angular displacement θ is provided in degrees and must first be converted to radians:

$$\theta = \text{radians(Angle)} \tag{64}$$

  We also define a fixed angle β based on the system geometry as:

$$\beta = \arcsin\left(\frac{r_s}{s_o}\right) \tag{65}$$

  Next, the angle α is computed from the complementary angles in the triangle formed by the actuator:

$$\alpha = \frac{\pi}{2} - \theta - \beta \tag{66}$$

- Linear Displacement Components

  To compute the current length of the system, we first calculate the horizontal and vertical components of the distance from the bottom actuator support to the axis of rotation. These are given by:

$$p_q = p_o \sin(\alpha) \tag{67}$$

$$q_o = p_o \cos(\alpha) \tag{68}$$

  The constant distance from the top actuator support to the axis of rotation is given by:

$$r_o = \frac{r_s}{\sin(\beta)} \tag{69}$$

  The horizontal distance between the top actuator support and the bottom actuator support is calculated as:

$$r_q = r_o - q_o \tag{70}$$

- Total Length Calculation

  The total length of the actuator at any given time, denoted as r_p, is computed using the Pythagorean theorem from the horizontal and vertical components:

$$r_p = \sqrt{r_q^2 + p_q^2} \tag{71}$$

- Strain Calculation

  The strain ε is defined as the relative change in the system's length compared to its original length, s_o. It is calculated as:

  $$\varepsilon = \frac{r_p - s_o}{s_o} \times 100 \tag{72}$$

  where the factor of 100 converts the strain to a percentage.

- Strain Rate Calculation

  The strain rate is calculated as the first derivative of strain with respect to time. Using a finite difference approximation, the strain rate between consecutive time points is given by:

  $$\text{Strain Rate} = \frac{\varepsilon_{i+1} - \varepsilon_i}{t_{i+1} - t_i} \tag{73}$$

  where t_i and t_{i+1} are consecutive time steps, and ε_i and ε_{i+1} are the corresponding strain values.

This approach allows us to compute both the strain and the strain rate of the system based on the measured angular displacement θ. The strain gives a measure of the relative elongation or compression of the system, while the strain rate describes how quickly the strain changes with time.

4.2.6 Isometric Tests

Isometric tests play a crucial role in evaluating the performance of soft actuators under constant strain conditions. During these tests, actuators are constrained to a fixed length, which allows for the measurement of force generated in response to varying internal pressures. This setup is essential for understanding how applied pressure influences force output, which directly impacts the actuator's functionality in practical applications.

The isometric tests were conducted with each actuator constrained at its non-contracted length. The results demonstrated an almost perfect linear correlation between force and pressure values, as shown in Figure 19. As expected, the slope of the force-pressure relationship ($\frac{dF}{dP}$) decreases with increasing stiffness, highlighting the dependency of force generation on the actuator's material stiffness.

This linear relationship indicates that the actuator respond predictably to changes in pressure, allowing force to be derived from pressure measurements alone, simplifying the process by eliminating the need for complex force sensors. Since pressure sensors are typically easier to integrate into systems, this method enhances practicality in applications.

The relationship between force and pressure can be expressed with the linear equation:

$$F = kP + c, \tag{74}$$

This formula allows us to calculate force based solely on pressure measurements.
From the experimental data obtained through linear approximation, the specific equation describing the force in relation to pressure is:

$$F = 0.0554 \cdot P + 1.89 \tag{75}$$

This means that for any given pressure P, the corresponding force F can be calculated effortlessly, removing the need for more complicated force measurement setups.

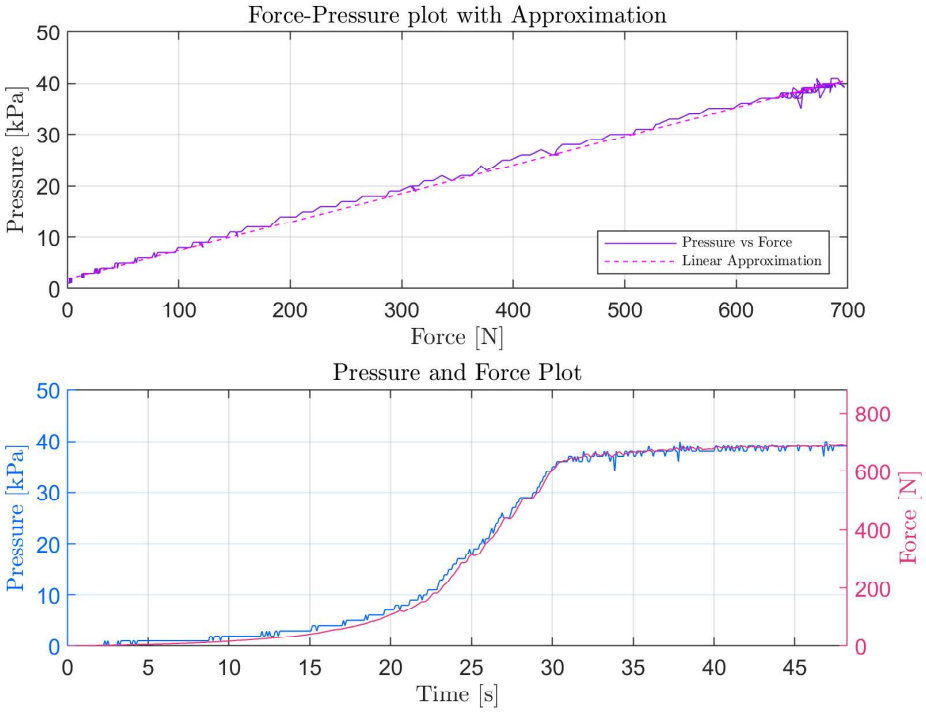


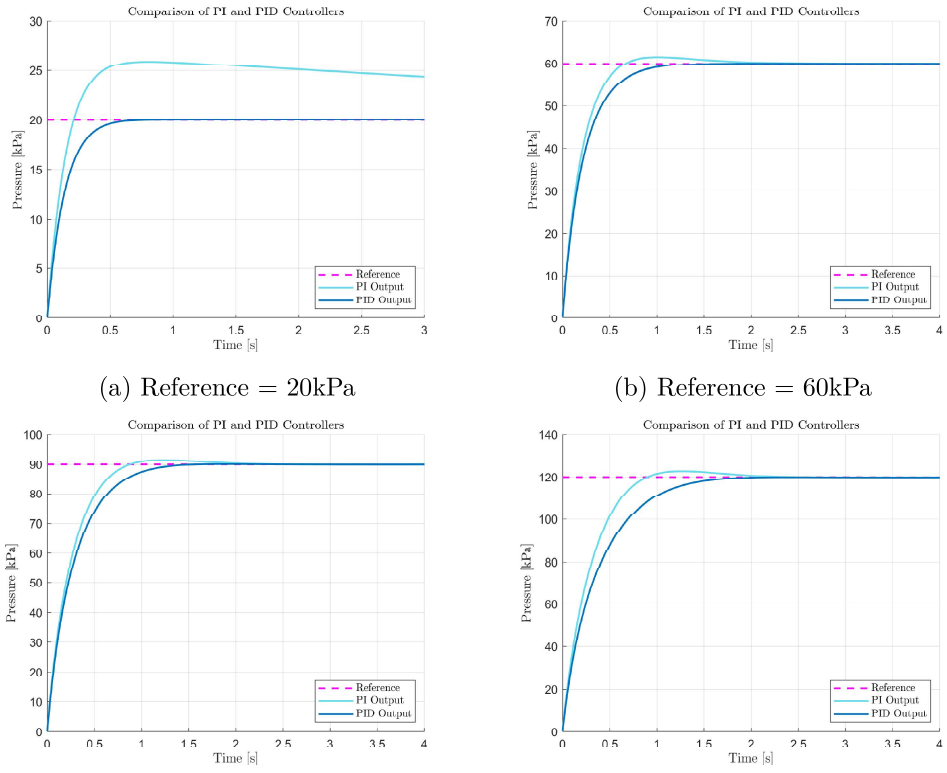
Figure 19: Isometric test measuring the force and pressure.

4.3 Simulation Experiments in Matlab

The aim of the Matlab simulation experiments was to design an optimal controller tailored to the model of the soft actuator. Several control strategies were evaluated, with the Proportional-Integral (PI) and Proportional-Integral-Derivative (PID) controllers delivering the most promising results in terms of balancing control effort and output accuracy. The key objectives of the simulation experiments were:

- To develop a controller optimized for the model of the soft actuator.
- To evaluate the performance of the controllers under simulated conditions.
- To ensure practical applicability of the controller in real-world environments.

However, discrepancies between the simulation results and real-world performance were observed, primarily due to imperfections in the model and the inherent variability of the actuators.



(a) Reference = 20kPa

(b) Reference = 60kPa

(c) Reference = 90kPa

(d) Reference = 120kPa

Figure 20: Matlab simulation of PI and PID controllers

To address this, numerous simulations were conducted to refine the controller, minimizing overshoot and ensuring stable and accurate tracking of the reference pressure. The results, shown in Figure 20, indicate that the PID controller outperforms the PI controller, especially in terms of speed and overshoot minimization. The PID demonstrates greater precision at lower pressures, such as 20kPa, while at higher pressures (e.g., 90kPa and above), the difference in response time between the two controllers is negligible.
The improved performance of the PID controller at lower pressures can be attributed to its derivative term, which is added only in the low-pressure range (below 25kPa). This makes the PID more responsive and precise in these scenarios compared to the PI controller, since PI controller, lacking this derivative action, is less responsive to sudden changes in pressure, especially at lower values.

These simulations focused solely on evaluating the PI and PID controllers designed from the model, excluding other control strategies like On-Off or experimentally tuned PI controllers. The goal was to assess the performance of the model-based controllers on real hardware, rather than optimizing experimental controllers through simulation.

4.4 Integrated Hardware Experiments

The final phase of experimentation focused on validating the reliability and performance of various controllers implemented for the soft actuator system. These experiments were critical in assessing how well controllers, developed through both experimental tuning and simulations, performed on real hardware.
Several key parameters were considered to ensure accuracy and consistency during testing:

- **Pressure Sensor:** A high-precision pressure sensor was used to provide accurate feedback on the internal pressure of the actuator, which is essential for evaluating controller performance.
- **Maximum Set Pressure:** A maximum pressure of 130 kPa was imposed to avoid over-pressurizing the actuator and maintain operational safety.
- **Power Limitation:** Input power was capped at 35 W to ensure consistency across all tests and to prevent actuator overheating.

4.4.1 Isometric Test with fixed pressure

The hardware experiments were conducted under the same conditions (as far as possible) with different reference pressures for each test. The set-up was characterized by:

- A pressure sensor was adopted for providing feedback on the acquired pressure data.

- A power supply was used to control power through current and voltage regulation.
- An Arduino Uno microcontroller was connected for data acquisition and control purposes.
- The actuator was constrained at its non-contracted and relaxed length (0% strain).
- The same actuator was utilized for all the experiments.

These components were integrated to enable real-time feedback control and comparison of various controllers under identical experimental conditions. The following control strategies were tested:

- **On-Off Control:** A simple control scheme that switches between maximum power and no power based on the pressure deviation from the set point.
- **PI Controller (Experimental Tuning):** A Proportional-Integral controller experimentally tuned on real hardware to balance response speed and system stability, minimizing overshoot.
- **PI Controller (Simulation-Based):** A Proportional-Integral controller developed through simulation from the actuator model. This allowed for a direct comparison between experimental tuning and simulation-based design.
- **Adaptive PID Controller (Simulation-Based):** An Adaptive Proportional-Integral-Derivative controller developed in simulation, with the ability to adjust its parameters dynamically in response to changing actuator behavior, especially useful for handling the nonlinearities typical of soft actuators.

The experiments were designed to evaluate key performance metrics of each controller, including:

- **Response Time:** The time taken to reach the desired pressure setpoint.
- **Stability:** The controller's ability to maintain the target pressure without oscillations or drift.
- **Accuracy:** The precision with which each controller could achieve and hold the set pressure.
- **Robustness:** The controller's performance when subjected to disturbances or variations in actuator characteristics due to nonlinearities or environmental factors.

Comparative results for each controller at various reference pressures (20kPa, 60kPa, 90kPa, and 130kPa) are shown in Figures 21 through 24. These figures illustrate the performance of each controller in terms of speed, stability, and precision in reaching the target pressure.
Key observations from the hardware experiments include:

- The **On-Off controller** exhibited significant oscillations around the set pressure, demonstrating its limitations in achieving stable control for this application.
- The **PI controller (experimental)** provided better stability but was slower in reaching the set pressure compared to the simulation-based PI controller.
- The **PI controller (simulation-based)** achieved faster response times, though slight overshoot occurred under certain conditions, likely due to inaccuracies between the simulated model and the real hardware.
- The **Adaptive PID controller** offered the best overall performance, quickly reaching the desired pressure with minimal overshoot and adapting to small changes in actuator behavior. This controller's ability to handle the inherent nonlinearities of soft actuators was particularly beneficial.

These tests provided crucial insights into the practical performance of the controllers, particularly in the context of soft actuators, which are characterized by nonlinear behavior and variability due to material properties and manufacturing inconsistencies. Simulation-based controllers, while useful, often need to be validated through hardware experiments to account for the following factors:

- **Nonlinearities in Soft Actuators:** Soft actuators exhibit varying stiffness, hysteresis, and other nonlinear behaviors. Simulations must accurately model these characteristics to ensure proper control in real-world scenarios.
- **Model-Environment Mismatch:** Although simulations provide valuable insights, discrepancies between the simulated environment and real-world hardware can cause performance gaps, underscoring the importance of hardware validation.
- **Adaptability of Controllers:** Adaptive control strategies, like the adaptive PID controller, excel in managing real hardware by adjusting to uncertainties and changes in actuator behavior over time.

The experimental results validated the effectiveness of simulation-based control designs but also emphasized the importance of real-world testing, especially for soft robotic systems. In particular, the Adaptive PID controller demonstrated superior robustness and adaptability, making it an ideal choice for managing the complex dynamics of soft actuators.

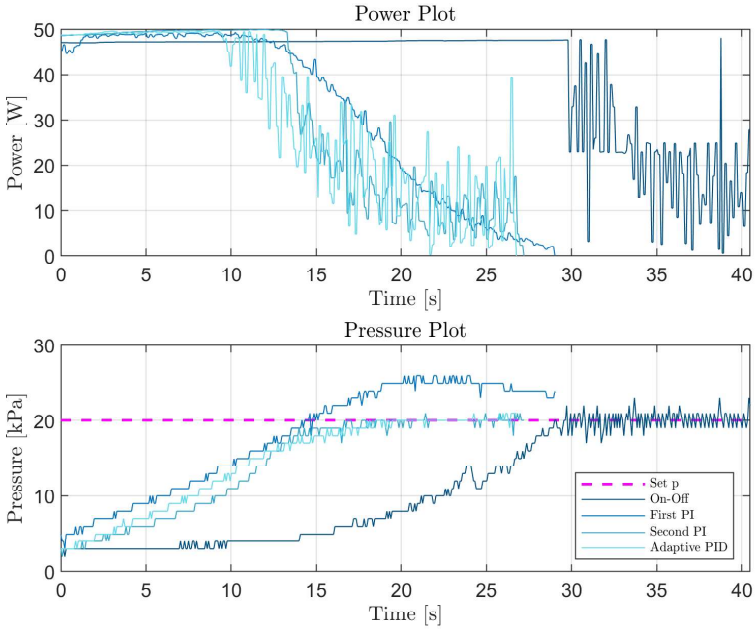


Figure 21: 20kPa reference pressure: Results comparison of Controllers

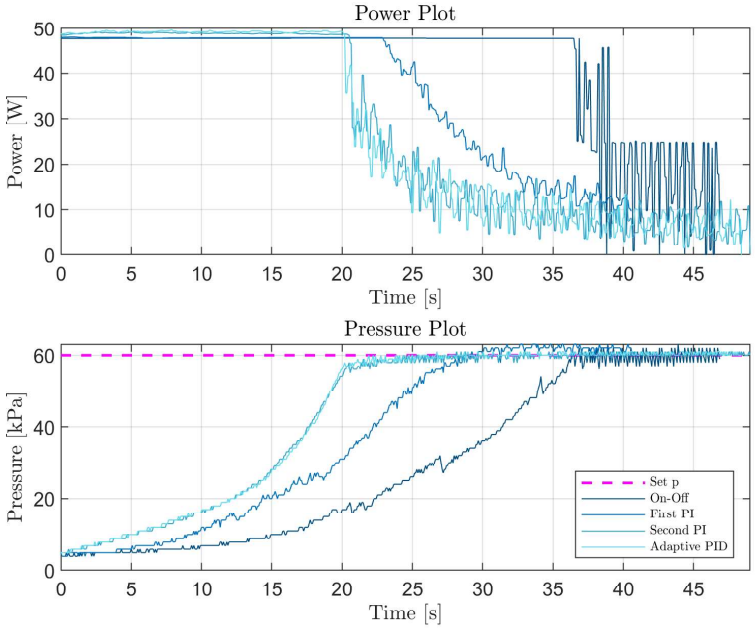


Figure 22: 60kPa reference pressure: Results comparison of Controllers

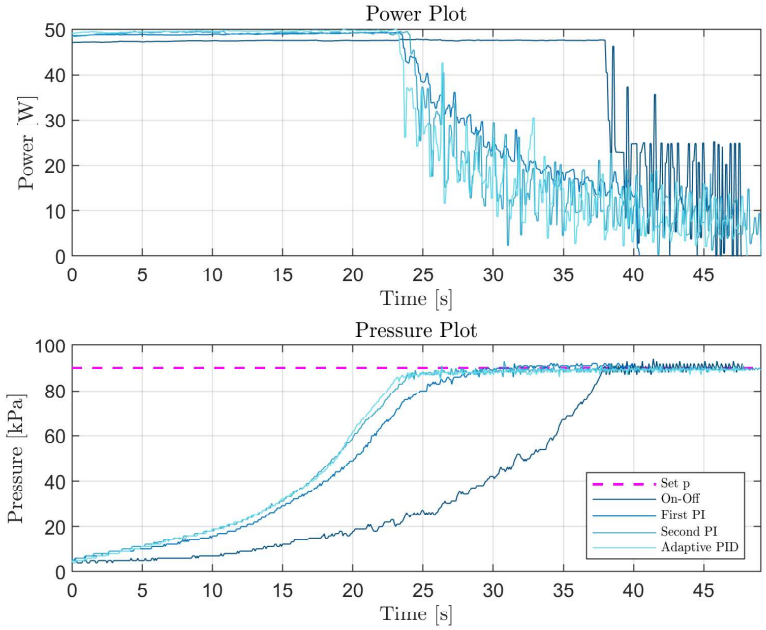


Figure 23: 90kPa reference pressure: Results comparison of Controllers

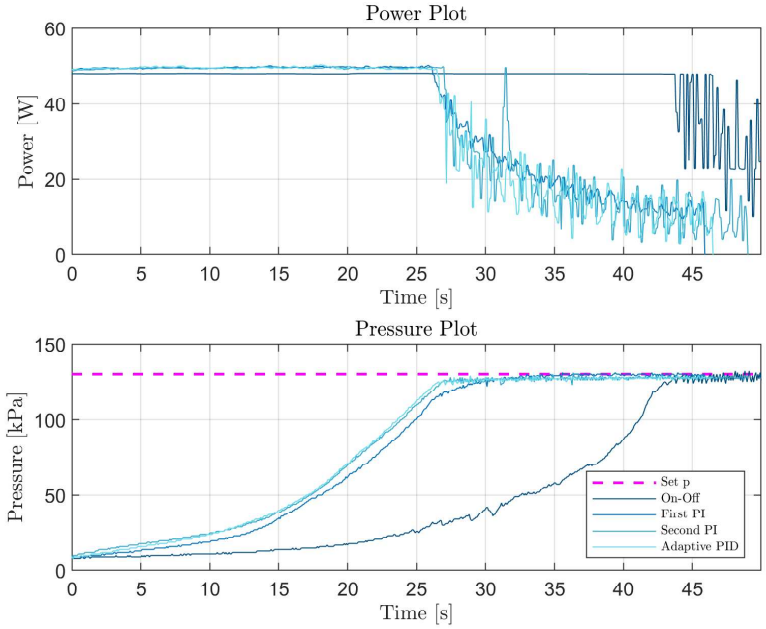


Figure 24: 130kPa reference pressure: Results comparison of Controllers

4.4.2 Isometric Test with Variable Pressure

In addition to the static pressure tests, further experiments were conducted to evaluate the controllers' stability and responsiveness under varying pressure conditions. In these tests, the actuator was subjected to the same experimental setup as described earlier, but with a dynamic reference pressure that increased of 10 kPa from 20 kPa until 130 kPa, the maximum limit imposed for safety reasons.

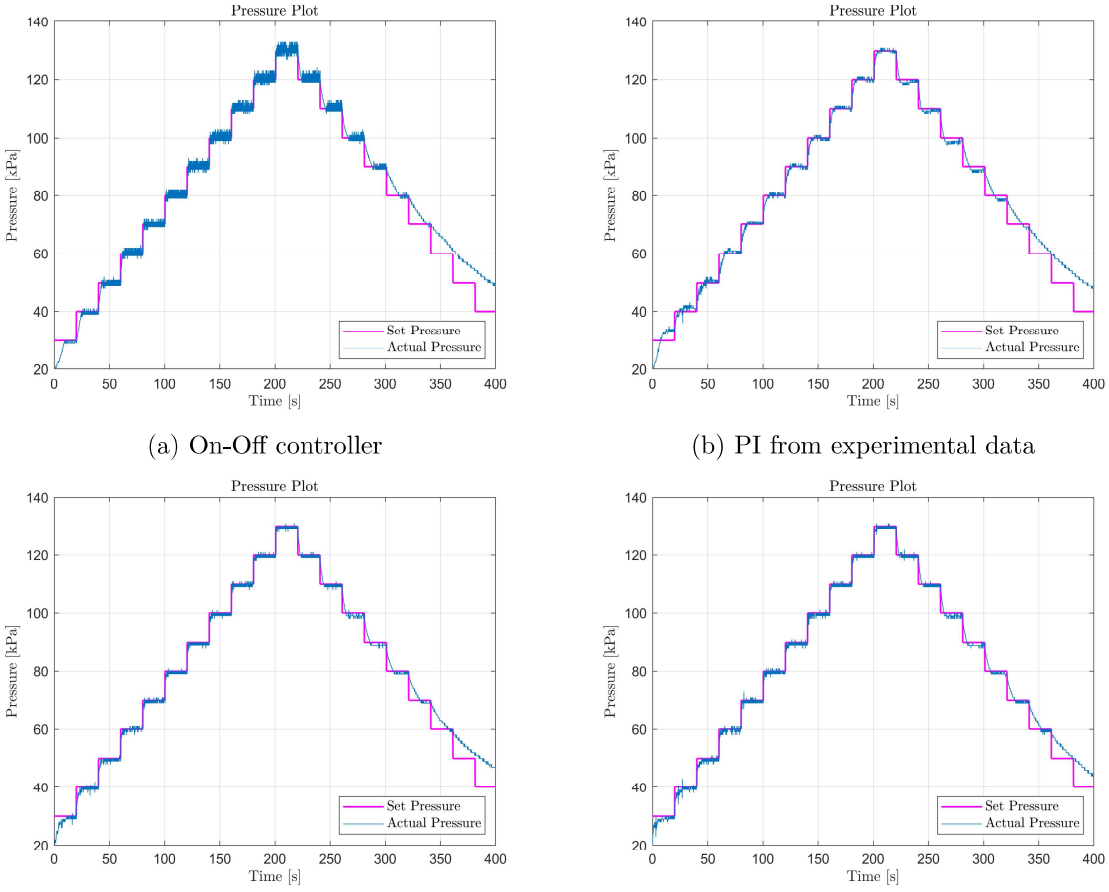
The results of these experiments are presented in Figure 25. One of the most evident differences lies between the On-Off controller and the other control strategies: the On-Off exhibited significantly more oscillations, with pressure deviations of approximately ± 3 kPa around the reference points, while both the PI controllers (from experimental and simulation data) and the adaptive PID controller demonstrated greater stability, with oscillations limited to around ± 1 kPa.

Another key observation is the performance of the PI controller tuned with experimental data, in fact this controller exhibited more pronounced overshoot compared to the PI derived from simulations and the adaptive PID controller. The larger overshoot not only affected the precision but also resulted in a slower response time, particularly at lower pressures (below 50 kPa), therefore both the simulated PI controller and the adaptive PID controller achieved the target pressure more quickly and with less overshoot.

When comparing the simulated PI controller and the adaptive PID controller, there was little difference in terms of speed or precision; however it is important to account for potential variability in the experiments, as well as external sources of error that could impact performance metrics. In real-world testing environments, factors such as slight inconsistencies in actuator manufacturing and environmental conditions may introduce variability in results, making robust evaluation essential.

Finally, it is worth noting that the controllers responded differently when the reference pressure was increased compared to when it was decreased: in particular the pressure increase (i.e. the "upstairs" phase) was precise and rapid across all controllers while the pressure decrease (i.e. the "downstairs" phase) was notably slower, which can be attributed to the inherent physical limitations of the soft actuator system, as discussed in Section 4.2.4. These limitations were consistent across all controllers, leading to a comparable performance in this phase of the tests.

These experiments highlight the superior performance of the simulation-derived PI and adaptive PID controllers over the On-Off and experimentally tuned PI controller and specifically the adaptive PID controller proved to be the most effective in maintaining stability and accuracy under dynamic pressure conditions, making it a highly suitable choice for controlling soft actuators in real-time applications.



(a) On-Off controller

(b) PI from experimental data

(c) PI from simulated data

(d) Adaptive PID

Figure 25: Experiment with variable reference pressure comparing different controllers.

4.4.3 Robustness Test

The final experiment was conducted to evaluate the robustness of the different controllers when applied to soft actuators with varying physical characteristics. Given that the soft actuators used in this study are handcrafted, inconsistencies in their manufacturing—such as differences in silicone tube thickness, coil length, and water content—can result in variability in performance. This test was aimed at determining how well each controller adapts to such variations, which are unavoidable due to the unreliable nature of manual production processes.
To assess robustness, the controllers were tested on a second set of soft actuators that exhibited slightly different properties compared to the ones used in earlier experiments. Fig. 26 illustrates the performance comparison of these controllers, measuring both the pressure response and the corresponding angle of deformation achieved by the actuator at a reference pressure of 90 kPa. The key observations from this experiment include:

- **Consistency Across Actuators:** Despite the inherent variability in the soft actuators, the relative performance hierarchy of the controllers remained consistent. The adaptive PID controller consistently exhibited the fastest response across both the original and the modified actuators.
- **Speed of Response:** The adaptive PID controller was the fastest in reaching the reference pressure in both actuators. This can be attributed to its ability to adjust its parameters dynamically, making it more effective in compensating for actuator variations and system nonlinearities. The PI controllers (both experimental and simulation-based) performed slower than the PID, with the experimentally-tuned PI controller being the least responsive.
- **Handling Variability:** The On-Off controller demonstrated significant oscillations and was unable to maintain the desired pressure steadily, particularly with the new actuators. This highlights its limitations in adapting to actuator variability, which is a common characteristic of soft robotics systems.
- **Precision and Stability:** While all controllers were able to eventually achieve the target pressure, the adaptive PID maintained greater precision with minimal overshoot, even with the variations in actuator properties. The PI controllers exhibited moderate overshoot, particularly the experimentally-tuned one, which struggled with both speed and accuracy.

These findings underscore the importance of using adaptive control strategies, particularly in systems where hardware variability is a factor. The adaptive PID controller's ability to adjust to changing dynamics and compensate for uncertainties makes it a superior choice for real-world applications involving soft actuators. In contrast, traditional controllers,

such as PI or On-Off, may require frequent retuning or exhibit suboptimal performance when dealing with actuator variability.

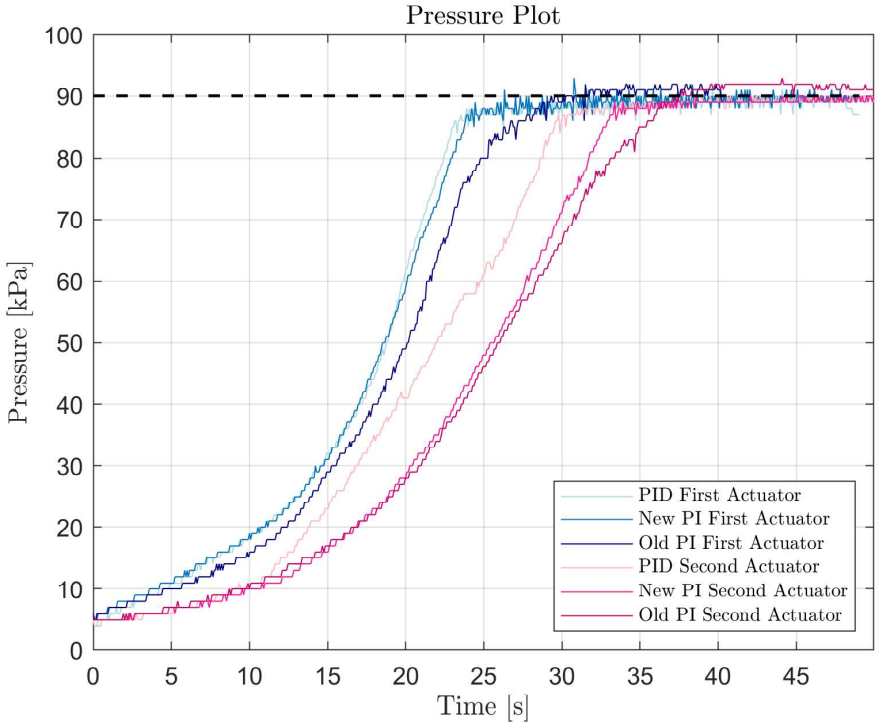


Figure 26: Robustness test comparing controller performance across different soft actuators at a reference pressure of 90kPa.

In conclusion, the robustness test confirms that while all controllers can achieve the target pressure, adaptive control techniques are essential when dealing with the inherent variability of soft actuators. The ability to dynamically adjust control parameters enables more reliable and efficient performance, especially in unpredictable real-world environments where actuator characteristics may change due to factors such as wear, environmental conditions, or inconsistent manufacturing.

5 Conclusions and Future Works

This thesis explored the development of effective control strategies for soft actuators, aiming to achieve an optimal balance between accuracy, speed, and computational efficiency. The inherent non-linearities and variability of soft actuators present unique challenges in the design of control and modeling systems. The primary focus was on three conventional control strategies: On-Off, Proportional-Integral (PI), and Proportional-Integral-Derivative (PID) controllers.

The main objectives were to design an optimal controller for soft actuators and to ensure practical applicability in real-world environments. Simulations indicated that both the PI and PID controllers provided satisfactory performance metrics regarding speed and precision, with the PID controller demonstrating superior effectiveness, particularly in achieving rapid response times with minimal overshoot, supporting the use of PID controllers in soft robotics applications.

Extensive simulations and hardware experiments were conducted to evaluate various control strategies, including Nonlinear Model Predictive Control (NMPC) and Reinforcement Learning (RL), which have been noted for their potential in soft robotics. However, real hardware experimentation revealed that implementing these advanced strategies would be counterproductive. In fact the RL approach relies on optimizing control strategies through extensive training phases, which are computationally expensive and time-consuming. Similarly, the NMPC would require significant computational resources, which are not justifiable given the limited potential for performance improvement due to the actuator's physical constraints.

Both PI and PID controllers performed significantly worse in real-life conditions compared to their simulation results, indicating that the introduction of RL would introduce minimal enhancements in the speed because the maximum speed of the soft actuator is influenced not only by the control strategy but also by its physical properties. This limitation implies that investing in RL or NMPC, despite their promise of improved precision, would result in marginal gains that do not justify the extensive computational effort. Therefore, the PID controller emerged as the optimal choice for the application, effectively balancing performance while minimizing computational demands.

In experiments, the PI controller exhibited slower response times and more significant overshoot compared to the PID controller. These insights emphasize the importance of selecting the appropriate controller for specific applications within soft robotics. Although both controllers achieved the target pressure, the PID controller's ability to adaptively

manage variations in the actuator behaviour provided a distinct advantage.

The robustness of the controllers against variations in actuator properties, common in handcrafted soft actuators, was also assessed. The adaptive capabilities of the PID controller maintained performance across different actuator configurations, demonstrating its effectiveness in real-world scenarios, while the On-Off control strategy showed significant limitations, resulting in oscillatory behaviour and poor stability, reinforcing the need for more sophisticated control techniques in soft robotics.

In conclusion, the work successfully demonstrated that traditional control strategies, particularly the PID controller, are optimal for managing soft actuators under various operational conditions. The results validate the effectiveness of simulation-based controller designs while highlighting the necessity of real-world testing to account for the complexities of soft actuator dynamics. Future research could explore hybrid control strategies that integrate elements of adaptive control with traditional methods to enhance performance further.

Additionally, investigating the application of machine learning techniques may refine control strategies and improve adaptability without the prohibitive training costs associated with RL. This thesis provides a foundation for further exploration into soft actuator control, emphasizing the importance of practical applicability and the critical evaluation of emerging methodologies within the field of soft robotics.

References

[1] Athar Ali et al. "Actuator and Contact Force Modeling of an Active Soft Brace for Scoliosis". In: *Bioengineering* 9.7 (2022). ISSN: 23065354. DOI: 10.3390.

[2] Samuel Alves et al. "Integrated Design Fabrication and Control of a Bioinspired Multimaterial Soft Robotic Hand". In: *Cyborg and Bionic Systems* 4 (2023). ISSN: 26927632. DOI: 10.34133/cbsystems.0051.

[3] Nazek El-Atab et al. "Soft Actuators for Soft Robotic Applications: A Review". In: *Advanced Intelligent Systems* 2.10 (2020). ISSN: 2640-4567. DOI: 10.1002/aisy.202000128.

[4] Yoseph Bar-Cohen. "Electroactive polymers as artificial muscles - Reality and challenges". In: *19th AIAA Applied Aerodynamics Conference* c (2001). DOI: 10.2514/6.2001-1492.

[5] Yoseph Bar-cohen. "Electroactive polymer (EAP) actuators — background review". In: *Mechanics of Soft Materials* 1 (2019). DOI: 10.1007/s42558-019-0005-1.

[6] Nicholas W. Bartlett et al. "A 3D-printed, functionally graded soft robot powered by combustion". In: *Science* 349.6244 (2015). ISSN: 10959203. DOI: 10.1126/science.aab0129.

[7] Antonio Bicchi. *Encyclopedia of Robotics*. Ed. by Marcelo H. Ang, Oussama Khatib, and Bruno Siciliano. August. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020. ISBN: 978-3-642-41610-1. DOI: 10.1007/978-3-642-41610-1.

[8] David Bombara, Steven Fowzer, and Jun Zhang. "Compliant, Large-Strain, and Self-Sensing Twisted String Actuators". In: *Soft Robotics* 9.1 (2022). ISSN: 21695180. DOI: 10.1089/soro.2020.0086.

[9] Daniel Bruder et al. "Modeling and Control of Soft Robots Using the Koopman Operator and Model Predictive Control". In: *Robotics: Science and Systems* (2019). ISSN: 2330765X. DOI: 10.15607/RSS.2019.XV.060. arXiv: 1902.02827.

[10] Andrea Centurelli. "Closed-loop control of Soft-Robots using Reinforcement Learning". In: *Politecnico di Torino* (2021).

[11] Yufeng Chen et al. "Controlled flight of a microrobot powered by soft artificial muscles". In: *Nature* 575.7782 (2019). ISSN: 14764687. DOI: 10.1038/s41586-019-1737-7.

[12] Marcello Chiaberge, Matteo Cianchetti, and Cecilia Laschi. "Study and design of a bioinspired actuation system for a soft robotic total artificial heart Supervisors". In: (2018).

[13] Ching-Ping Chou and Blake Hannaford. "Measurement and modeling of McKibben pneumatic artificial muscles". In: *IEEE Transactions on Robotics and Automation* 12.1 (1996). ISSN: 1042296X. DOI: 10.1109/70.481753.

[14] Chia Ye Chu and Rita M. Patterson. "Soft robotic devices for hand rehabilitation and assistance: A narrative review". In: *Journal of NeuroEngineering and Rehabilitation* 15.1 (2018). ISSN: 17430003. DOI: 10.1186/s12984-018-0350-6.

[15] M. Cianchetti et al. "A new design methodology of electrostrictive actuators for bio-inspired robotics". In: *Sensors and Actuators, B: Chemical* 142.1 (2009). ISSN: 09254005. DOI: 10.1016/j.snb.2009.08.039.

[16] William Coral et al. "Design and assessment of a flexible fish robot actuated by shape memory alloys". In: (2018).

[17] "Fast and robust: Hexapedal robots via shape deposition manufacturing". In: *International Journal of Robotics Research* 21.10-11 (2002). ISSN: 02783649. DOI: 10.1177/027836490202101083 7.

[18] Diogo Fonseca and Pedro Neto. "Force/Proximity hybrid sensors and Phase Change Actuators". In: *Universidade de Coimbra* (2021).

[19] Diogo Fonseca and Pedro Neto. "The untapped potential of electrically-driven phase transition actuators to power innovative soft robot designs". In: *Universidade de Coimbra* (2024).

[20] Kevin C. Galloway et al. "Soft Robotic Grippers for Biological Sampling on Deep Reefs". In: *Soft Robotics* 3.1 (2016). ISSN: 21695180. DOI: 10.1089/soro.2015.0019.

[21] Xiangyu Gao et al. "Piezoelectric Actuators and Motors: Materials, Designs, and Applications". In: *Advanced Materials Technologies* 5.1 (2020). ISSN: 2365709X. DOI: 10.1002/admt.201900716.

[22] Thomas George Thuruthel et al. "Control Strategies for Soft Robotic Manipulators: A Survey". In: *Soft Robotics* 5.2 (2018). ISSN: 21695180. DOI: 10.1089/soro.2017.0007.

[23] I.A. Gravagne, C.D. Rahn, and I.D. Walker. "Large deflection dynamics and control for planar continuum robots". In: *IEEE/ASME Transactions on Mechatronics* 8.2 (June 2003). ISSN: 1083-4435. DOI: 10.1109/TMECH.2003.812829.

[24] David A. Haggerty et al. "Control of soft robots with inertial dynamics". In: *Science Robotics* 8.81 (2023). ISSN: 24709476. DOI: 10.1126/scirobotics.add6864.

[25] Diego R. Higueras-Ruiz et al. "What is an artificial muscle? A comparison of soft actuators to biological muscles". In: *Bioinspiration Biomimetics* 17.1 (Jan. 2022). ISSN: 1748-3182. DOI: 10.1088/1748-3190/ac3adf.

[26] Zhujin Jiang, Yan Wang, and Ketao Zhang. "Development of a Pneumatically Actuated Quadruped Robot Using Soft – Rigid Hybrid Rotary Joints". In: (2024).

[27] Rianna Jitosho et al. "A Dynamics Simulator for Soft Growing Robots". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2021-May (2021). ISSN: 10504729. DOI: 10.1109/ICRA48506.2021.9561420. arXiv: 2011.01917.

[28] Rianna Jitosho et al. "Reinforcement Learning Enables Real-Time Planning and Control of Agile Maneuvers for Soft Robot Arms". In: *Proceedings of Machine Learning Research* 229.CoRL (2023). ISSN: 26403498.

[29] B.A. Jones and I.D. Walker. "Kinematics for multisection continuum robots". In: *IEEE Transactions on Robotics* 22.1 (Feb. 2006). ISSN: 1552-3098. DOI: 10.1109/TRO.2005.861458.

[30] Hunpyo Ju et al. "Closed-Loop Soft Robot Control Frameworks with Coordinated Policies Based on Reinforcement Learning and Proprioceptive Self-Sensing". In: *Advanced Functional Materials* 33.51 (2023). ISSN: 16163028. DOI: 10.1002/adfm.202304642.

[31] Nicholas Kellaris et al. "Peano-HASEL actuators: Muscle-mimetic, electrohydraulic transducers that linearly contract on activation". In: *Science Robotics* 3.14 (Jan. 2018). ISSN: 2470-9476. DOI: 10.1126/scirobotics.aar3276.

[32] Ozgun Kilic Afsar et al. "OmniFiber: Integrated Fluidic Fiber Actuators for Weaving Movement based Interactions into the 'Fabric of Everyday Life'". In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, Oct. 2021. ISBN: 9781450386357. DOI: 10.1145/3472749.3474802.

[33] Jaehwan Kim et al. "Review of Soft Actuator Materials". In: *International Journal of Precision Engineering and Manufacturing* 20.12 (2019). ISSN: 20054602. DOI: 10.1007/s12541-019-00255-1.

[34] Meng Li et al. "Soft actuators for real-world applications". In: *Nature Reviews Materials* 7.3 (Nov. 2021). ISSN: 2058-8437. DOI: 10.1038/s41578-021-00389-7.

[35] Shuguang Li et al. "Fluid-driven origami-inspired artificial muscles". In: *Proceedings of the National Academy of Sciences* 114.50 (Dec. 2017). ISSN: 0027-8424. DOI: 10.1073/pnas.1713450114.

[36] Yingqi Li, Xiaomei Wang, and Ka Wai Kwok. "Towards Adaptive Continuous Control of Soft Robotic Manipulator using Reinforcement Learning". In: *IEEE International Conference on Intelligent Robots and Systems* 2022-Octob (2022). ISSN: 21530866. DOI: 10.1109/IROS47612.2022.9981335.

[37] Márcio D. Lima et al. "Electrically, Chemically, and Photonically Powered Torsional and Tensile Actuation of Hybrid Carbon Nanotube Yarn Muscles". In: *Science* 338.6109 (Nov. 2012). ISSN: 0036-8075. DOI: 10.1126/science.1226762.

[38] Huai Ti Lin, Gary G. Leisk, and Barry Trimmer. "GoQBot: A caterpillar-inspired soft-bodied rolling robot". In: *Bioinspiration and Biomimetics* 6.2 (2011). ISSN: 17483182. DOI: 10.1088/1748-3182/6/2/026007.

[39] Hod Lipson. "Challenges and Opportunities for Design, Simulation, and Fabrication of Soft Robots". In: *Soft Robotics* 1.1 (Mar. 2014). ISSN: 2169-5172. DOI: 10.1089/soro.2013.0007.

[40] Xuan Liu, Cagdas D. Onal, and Jie Fu. "Reinforcement Learning of CPG-Regulated Locomotion Controller for a Soft Snake Robot". In: *IEEE Transactions on Robotics* 39.5 (2023), pp. 3382–3401. ISSN: 19410468. DOI: 10.1109/TRO.2023.3286046. arXiv: 2207.04899.

[41] Alexandro López-González, Juan C. Tejada, and Janet López-Romero. "Review and Proposal for a Classification System of Soft Robots Inspired by Animal Morphology". In: *Biomimetics* 8.2 (2023). ISSN: 23137673. DOI: 10.3390/biomimetics8020192.

[42] Carmel Majidi. "Soft Robotics: A Perspective - Current Trends and Prospects for the Future". In: *Soft Robotics* 1.1 (2014). ISSN: 21695180. DOI: 10.1089/soro.2013.0001.

[43] Y. Mamiya. "Applications of Piezoelectric Actuator". In: *Nec Technical Journal* 1.5 (2006).

[44] Jesus Marquez et al. "Hardware-in-the-Loop Soft Robotic Testing Framework Using an Actor-Critic Deep Reinforcement Learning Algorithm". In: *IEEE Robotics and Automation Letters* 8.9 (2023). ISSN: 23773766. DOI: 10.1109/LRA.2023.3301215.

[45] Safeh Clinton Mawah and Yong Jai Park. "Tendon-Driven Variable-Stiffness Pneumatic Soft Gripper Robot". In: *Robotics* 12.5 (2023). ISSN: 22186581. DOI: 10.3390/robotics12050128.

[46] Hamish McGorm et al. "Turning Up the Heat: An Evaluation of the Evidence for Heating to Promote Exercise Recovery, Muscle Rehabilitation and Adaptation". In: *Sports Medicine* 48.6 (2018). ISSN: 11792035. DOI: 10.1007/s40279-018-0876-6.

[47] Yiit Mengüç et al. "Wearable soft sensing suit for human gait measurement". In: *International Journal of Robotics Research* 33.14 (2014). ISSN: 17413176. DOI: 10.1177/0278364914543793.

[48] Miriyev. "A Focus on Soft Actuation". In: *Actuators* 8.4 (2019).

[49] Aslan Miriyev, Kenneth Stack, and Hod Lipson. "Soft material for soft actuators". In: *Nature Communications* 8.1 (Sept. 2017). ISSN: 2041-1723. DOI: 10.1038/s41467-017-00685-3.

[50] Seyed M. Mirvakili et al. “Actuation of untethered pneumatic artificial muscles and soft robots using magnetically induced liquid-to-gas phase transitions”. In: *Science Robotics* 5.41 (Apr. 2020). ISSN: 2470-9476. DOI: 10.1126/scirobotics.aaz4239.

[51] Rafael Correia Molter. “Toward bio-inspired artificial muscle fascicles based on liquid crystal elastomer”. In: September (2023).

[52] Ryota Morimoto et al. “Model-free reinforcement learning with ensemble for a soft continuum robot arm”. In: *2021 IEEE 4th International Conference on Soft Robotics, RoboSoft 2021* (2021). DOI: 10.1109/RoboSoft51838.2021.9479340.

[53] Noel Naughton et al. “Elastica: A Compliant Mechanics Environment for Soft Robotic Control”. In: *IEEE Robotics and Automation Letters* 6.2 (Apr. 2021). ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3063698. arXiv: 2009.08422.

[54] Krista Niebaum, Laurie McCauley, and Carolina Medina. “Rehabilitation Physical Modalities”. In: *Canine Sports Medicine and Rehabilitation: Second Edition* (2018). DOI: 10.1002/9781119380627.

[55] Xinjian Niu et al. “INVESTIGATION of ROBOTIC BRACES of PATIENTS with IDIOPATHIC SCOLIOSIS (IS) - REVIEW of the LITERATURE and DESCRIPTION of A NOVEL BRACE”. In: *Journal of Mechanics in Medicine and Biology* 18.8 (2018). ISSN: 02195194. DOI: 10.1142/S0219519418400389.

[56] Amir Pagoli et al. “Review of soft fluidic actuators: Classification and materials modeling analysis”. In: *Smart Materials and Structures* 31.1 (2022). ISSN: 1361665X. DOI: 10.1088/1361-665X/ac383a.

[57] Gianluca Palli et al. “Modeling and control of the twisted string actuation system”. In: *IEEE/ASME Transactions on Mechatronics* 18.2 (2013). ISSN: 10834435. DOI: 10.1109/TMECH.2011.2181855.

[58] Min Pan et al. “Soft Actuators and Robotic Devices for Rehabilitation and Assistance”. In: *Advanced Intelligent Systems* 4.4 (2022). ISSN: 2640-4567. DOI: 10.1002/aisy.202100140.

[59] Yong Lae Park et al. “Design and control of a bio-inspired soft wearable robotic device for ankle-foot rehabilitation”. In: *Bioinspiration and Biomimetics* 9.1 (2014). ISSN: 17483182. DOI: 10.1088/1748-3182/9/1/016007.

[60] Jingyang Peng and Xiongbiao Chen. “A Survey of Modeling and Control of Piezoelectric Actuators”. In: *Modern Mechanical Engineering* 03.01 (2013). ISSN: 2164-0165. DOI: 10.4236/mme.2013.31001.

[61] Panagiotis Polygerinos et al. “Soft robotic glove for combined assistance and at-home rehabilitation”. In: *Robotics and Autonomous Systems* 73 (2015). ISSN: 09218890. DOI: 10.1016/j.robot.2014.08.014.

[62] Steven I. Rich, Robert J. Wood, and Carmel Majidi. "Untethered soft robotics". In: *Nature Electronics* 1.2 (2018). ISSN: 25201131. DOI: 10.1038/s41928-018-0024-1.

[63] Val J. Robertson, Alex R. Ward, and Peter Jung. "The effect of heat on tissue extensibility: A comparison of deep and superficial heating". In: *Archives of Physical Medicine and Rehabilitation* 86.4 (Apr. 2005). ISSN: 00039993. DOI: 10.1016/j.apmr.2004.07.353.

[64] Ellen T. Roche et al. "A bioinspired soft actuated material". In: *Advanced Materials* 26.8 (2014). ISSN: 09359648. DOI: 10.1002/adma.201304018.

[65] Ellen T. Roche et al. "Soft robotic sleeve supports heart function". In: *Science Translational Medicine* 9.373 (2017). ISSN: 19466242. DOI: 10.1126/scitranslmed.aaf3925.

[66] Matthias Rolf and Jochen J. Steil. "Constant curvature continuum kinematics as fast approximate model for the Bionic Handling Assistant". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. October 2012. IEEE, Oct. 2012. ISBN: 978-1-4673-1736-8. DOI: 10.1109/IROS.2012.6385596.

[67] Daniela Rus and Michael T. Tolley. "Design, fabrication and control of soft robots". In: *Nature* 521.7553 (2015). ISSN: 14764687. DOI: 10.1038/nature14543.

[68] Michael T. Tolley et al. "A Resilient, Untethered Soft Robot". In: *Soft Robotics* 1.3 (Sept. 2014). ISSN: 2169-5172. DOI: 10.1089/soro.2014.0008.

[69] Bertrand Tondu. "Towards a theory of actuators: A new classification proposal for actuation systems". In: *Sensors and Actuators, A: Physical* 289 (2019). ISSN: 09244247. DOI: 10.1016/j.sna.2019.02.036.

[70] Deepak Trivedi, Amir Lotfi, and C.D. Rahn. "Geometrically Exact Models for Soft Robotic Manipulators". In: *IEEE Transactions on Robotics* 24.4 (Aug. 2008). ISSN: 1552-3098. DOI: 10.1109/TRO.2008.924923.

[71] Ryan L. Truby et al. "Soft Somatosensitive Actuators via Embedded 3D Printing". In: *Advanced Materials* 30.15 (2018). ISSN: 15214095. DOI: 10.1002/adma.201706383.

[72] Alex Villanueva, Colin Smith, and Shashank Priya. "A biomimetic robotic jellyfish (Robojelly) actuated by shape memory alloy composite actuators". In: *Bioinspiration Biomimetics* 6.3 (Sept. 2011). ISSN: 1748-3182. DOI: 10.1088/1748-3182/6/3/036004.

[73] Jue Wang and Alex Chortos. "Control Strategies for Soft Robot Systems". In: *Advanced Intelligent Systems* 4.5 (2022). ISSN: 2640-4567. DOI: 10.1002/aisy.202100165.

[74] Michael Wehner et al. “An integrated design and fabrication strategy for entirely soft, autonomous robots”. In: *Nature* 536.7617 (Aug. 2016). ISSN: 0028-0836. DOI: 10.1038/nature19100.

[75] Yang Yang et al. “Yang Yang , Member , IEEE , Zicheng Kan , Yazhan Zhang , Yu Alexander Tse , and Michael Yu Wang ”. In: (2019), pp. 3983–3989.

[76] Michael C. Yip and Gunter Niemeyer. “On the Control and Properties of Supercoiled Polymer Artificial Muscles”. In: *IEEE Transactions on Robotics* 33.3 (June 2017). ISSN: 1552-3098. DOI: 10.1109/TRO.2017.2664885.

[77] Jun Zhang et al. “Robotic Artificial Muscles: Current Progress and Future Perspectives”. In: *IEEE Transactions on Robotics* 35.3 (2019). ISSN: 19410468. DOI: 10.1109/TRO.2019.2894371.

[78] Ali Zolfagharian et al. “Evolution of 3D printed soft actuators”. In: *Sensors and Actuators, A: Physical* 250 (2016). ISSN: 09244247. DOI: 10.1016/j.sna.2016.09.028.