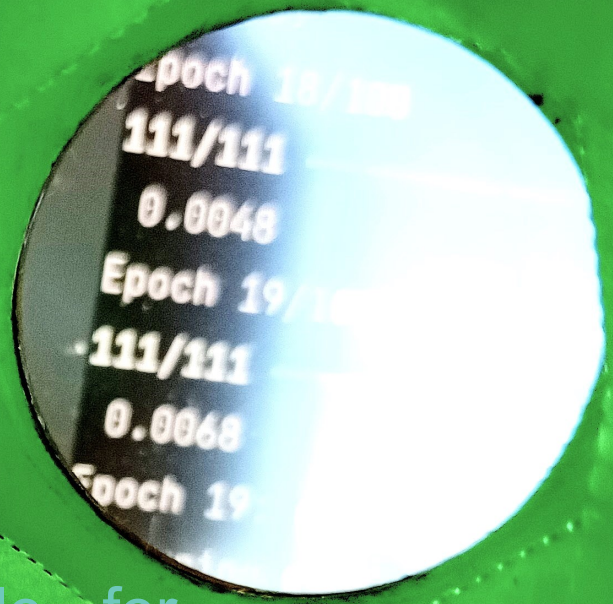
Technische Universiteit Delft

Lifelong Learning for a Kresling Origami Robot



A Plug-and-play Module for Supervised Learning of Target Adaptations

E. De Vroey

TU Delft

Lifelong Learning for a Kresling Origami Robot

A Plug-and-play Module for Supervised Learning of Target Adaptations

by

E. De Vroey

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday October 25, 2024 at 9:00 AM.

Student number:	4685156	
Duration:	September 2023 - October 2024	
Thesis committee:	Dr. ing. J. Kober	TU Delft, supervisor
	Dr. M. Wiertlewski	TU Delft

TU Delft

Preface

When asked numerous times over the past year what I was doing my Master's Thesis on, I can remember thinking how lucky I was with the title of my topic. *"Lifelong Learning for a Kresling Origami Robot"* speaks to the imagination, and turns the person who asked the question–likely out of politeness–into a genuinely interested listener.

Upon finding this thesis topic amongst the other options, I was surprised that it was still available: Origami meets machine learning, and you get to use your *arts-and-crafts*-skills while designing and building your very own robot–surely the list of available options had not been updated yet. To my great relief, Professor Kober assured me that it was still free, and we sat down to discuss the prospects. I had recently finished an internship where I gained some hands-on experience in implementing machine learning by myself, and I like crafting and tinkering, so this topic seemed cut-out for me. That said, I would not have considered myself a "maker". I never worked with microcontrollers before, nor soldered my own circuit-boards together; I hardly really understood the basic principle of a breadboard. Thirteen months later, I can say that I do not regret choosing this topic.

I would therefore like to thank Professor Jens Kober for his encouragement and support throughout the entire process. Often finding myself at an impasse, Professor Kober would offer some valuable insight or alternative solution to look into, completely freeing me up to *get moving* again. Probably also indebted to Professor Kober for that same reason are my parents, whom I really need to thank for continually putting up with my repeated statements that *"yes, I really am nearly finished now"*. Their worries of me becoming an eternal student might have been a great deal worse, however, were it not for the kind people at the TU Delft Science Centre, who allowed me to access their MakerSpace. Without their lasercutter, the–to my own estimation–200 origami structures that I fabricated throughout the process would have likely taken months to cut out and construct.

Then there are my friends: I spent nearly the entirety of my thesis working in the Echo building, and without the assurance that Bram, Oscar, Thomas, Niels, and Kato would be there as well, all going through the same tough process, it would have been a lot harder to get up and go there every morning. Our so-called "Echo crew" was always ready to go for a spontaneous coffee break, to lunch together, or to grab a drink after a long day of work. On that note, I think I can spare a line of text to extend my gratitude to the volunteers at Architecture's faculty bar De BouwPub, who were always willing to provide said drinks. My housemates, both old and new, Tim, Frederique, Emiel, Tine, Loes, Maxim, Rune, Sien, and Nelson, I would like to thank for making each day more than just *work*. May the Bronx Blues Brothers soon have their musical breakthrough. Finally, I would like to thank Arnaud, who volunteered to proofread this attempt at a coherent work of scientific research–a thankless task which I hope he does not regret accepting.

With that said, I hope that you enjoy *"Lifelong Learning for a Kresling Origami Robot"* as much as I enjoyed the whole process of turning this concept from a mere title into a physical robot, and finally, into this document.

E. De Vroey
Delft, October 2024

Contents

Lifelong Learning for a Kresling Origami Robot:

A Plug-and-play Module for Supervised Learning of Target Adaptations

E. De Vroey

***Abstract*—Soft robots offer a variety of useful applications. However, the nature of their design makes them challenging to control using traditional techniques. Many applications therefore rely on machine learning-based methods, learning opaque control policies or dynamic models of the robot. This often results in overly complex and uninterpretable solutions for the problem at hand.**

In this paper, a simplified approach is considered: Using a common feedforward neural network as a *target adapter* module, learned adaptations are added to the desired targets that are fed to a simple PID controller, essentially "deceiving" it into a better performance. This adapter is completely separate from the controller itself, allowing for relatively simple controllers to control and adapt to complex and evolving systems.

The approach is tested in simulation on a simple mass-spring-damper control problem, where qualitative results show near-immediate improvement in controller performance. The approach is then tested on a demonstrator origami robot, which relies for its functionality on the dynamics of the Kresling origami spring. Results show that this simple adaptation approach is robust to changes in controller tuning and changes in system dynamics. However, the method is sensitive to the sampling frequency at which training data is recorded.

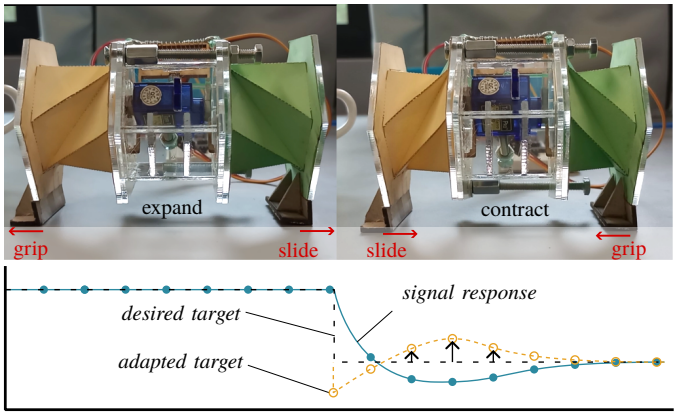


Fig. 1: The Kresling origami robot with target adaptation. *Top:* The "feet" of the robot grip when dragged backward, but slide when pushed forward, allowing the robot to move forward, also known as *two-anchor crawling* (Calisti et al., 2017). *Bottom:* The contraction target is adapted to reduce overshoot and unstable oscillations.

I. Introduction

Lifelong learning, the ability of a system to continuously adapt to new information throughout its lifetime, is a much sought-after feature for many applications. Rather than relying on policies that are "set in stone", these systems learn and evolve autonomously, changing and improving their behaviors as they encounter new tasks or environments. This ability makes lifelong learning a powerful tool for systems that operate in uncertain or evolving scenarios, where changes are common or to be expected. Not in the least for robotics.

Robots, like all mechanical systems, wear down over time—joints become more compliant, motors weaken, and sensors lose precision. Traditionally, maintaining a robot's performance might require manual software updates or recalibrations every so often (Lao et al., 2023). But as robots become more complex, especially in fields like soft robotics, where flexible, compliant materials introduce additional challenges, the need for autonomous adaptation becomes clear: By design, the materials used in the construction of soft robots are less sturdy and thus more prone to much faster deterioration, making manual updates or recalibrations impractical.

In addition, the complex system dynamics introduced by the unconventional robot configurations and use of materials make modeling their dynamics and control difficult. Deriving an analytical model for the dynamics of a soft robot can become so complicated to the point that it is impractical. Soft-robotics applications therefore heavily rely on the use of machine learning (ML) methods (Bhagat et al., 2019; Kim et al., 2021). These "black-box" techniques offer the possibility of modeling the robot's dynamics and allow for continuous adaptation but are often very opaque or complex, even though the task or environment setting of the soft robot could be something fairly simple.

This paper proposes a method of enhancing a traditional controller with ML abilities. The idea is that a basic controller can be tuned to a bare minimum sufficient performance, without worrying about the variations in dynamics introduced by soft robots. This controller is expanded with a separate module that takes care of the learning and adapting part of the challenge. The module then only needs to improve the performance of the controller, without requiring knowledge on the dynamics of the system, nor is the controller completely replaced by an uninterpretable "black-box" policy. The controller works in cooperation with the ML-module, separating the interpretable basic control from the opaque learning and adapting.

Techniques like parameter adaptation work much in the same way. An adapting or tuning module, quite often ML-based, can be "plugged" into a base controller, tuning its parameters *on the go* to achieve an optimal performance (Annaswamy & Fradkov, 2021; Landau et al., 2011). However, parameter adaptation requires knowledge on the architecture of the controller (e.g., how many, and which parameters should

be tuned) and such a module must thus be designed separately for each controller.

The proposed method tries to mitigate this downside by offering a "plug-and-play" module that can improve a controller's performance by simply adapting the desired target it receives (Figure 1, *bottom*). In this way, no previous knowledge on the type or tuning of the controller is required. In other words, as long as the fundamental assumptions of a controller are satisfied—i.e., it requires a state and a target as input and yields a control action—the method can remain completely agnostic to the specific controller being used.

As Tang and Wei (2022) point out, with soft robots offering many advantages in miniaturization, the physical space for controllers with heavy computational power can be limited. The "plug-and-play" module offers thus offers another advantage: externalizing the heavy computational task from the soft robot, which allows a (miniaturized) robot to just run a basic controller with low computational capabilities.

As a demonstration of the technique, this target adaptation approach is tested on a demonstrator origami robot, constructed in part from paper and which relies on the widespread Kresling pattern (Kresling, 2008) for its design (Figure 1, *top*). The Kresling origami spring (KOS) is characterized by its non-linear stiffness with deflection and bistable behavior (Masana & Daqaq, 2019), making it an interesting building block for many soft-robotics applications (Masana et al., 2024). The demonstrator robot is thus a good representation of a large cross-section in soft robot designs, as well as a useful playground for testing the adaptability of the method to more complex dynamics.

II. Related Work

This section will provide a useful background and context for the Kresling origami robot (Sections II-A and II-C) as well as a brief overview and comparison of related methods concerning lifelong learning (Section II-B).

A. Soft Robotics

Soft robotics is a field that has gained much interest over the past decade. Soft robots are constructed from deformable, soft materials which offer advantages in fields such as human-robot interaction, where their inherent compliance makes them safe to interact with. Other use-cases range from manipulating fragile objects, to medical treatment delivery inside the human body, and search-and-rescue (Hawkes et al., 2017; Kim et al., 2021; Ze et al., 2022).

Origami robots are a subset of soft robots. Inspired by the Japanese art of paper-folding, these robots rely on creases in their materials for both structural integrity and actuation methods. This allows for highly miniaturized designs that can self-deploy and can be actuated using affordable, lightweight mechanisms (Tang & Wei, 2022).

Of course, although the materials used often bend rather than break, they *are* prone to deteriorate faster than their rigid counterparts. Hence, combined with the aforementioned difficulty of modeling the dynamics, the usefulness of lifelong learning in this field becomes obvious (Kim et al., 2021). Additionally, the rapid deterioration and widely available materials make soft robots a fast and economical playground for testing lifelong learning techniques as well.

B. Lifelong Learning

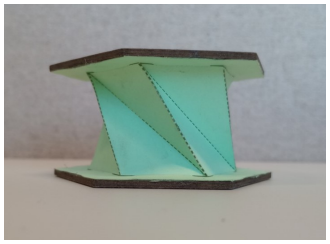
Many approaches to lifelong learning can be found in literature, with machine learning being an integral part in nearly all of them (Lesort et al., 2020). Broadly speaking, the approaches can be divided into four categories, which are not necessarily mutually exclusive: Methods that result in a "black-box" control policy, often obtained through reinforcement learning, where a policy is found by trying to optimize a *reward* function (Bhagat et al., 2019). Methods that model the dynamics of the system, to be used in combination with other controllers or to optimize an ML-based policy (e.g., Gillespie et al. (2018) and Thuruthel et al. (2019), respectively). Special architectures, such as the parameter adaptation or estimation mechanism, and iterative learning control, where a separate module capable of learning is combined with a base controller (Landau et al., 2011; R. J. Li & Han, 2005). Finally, the use of mechanisms that invoke some kind of memory, either by modifying the update method to be less destructive (e.g., *learning without forgetting* from Z. Li and Hoiem (2018)), or by implementing some form of library that can be accessed to store and retrieve information or parameters, as in e.g. Wang et al. (2020).

The techniques most similar to the method discussed in this paper are parameter adaptation/estimation and iterative learning control (ILC). In parameter adaptation, ML-based methods are used to directly tune the internal parameters of a controller. The various techniques used to achieve this rely on observing the interactions of the controller with the system or having access to the feedback signal of the controller (Landau et al., 2011). Furthermore, the estimation mechanism must be designed specifically for the parameters of the controller in question. Iterative learning control on the other hand only uses the error signal to improve a controller's performance. It achieves this by changing either the controller's reference signal (i.e., the target) or its control action, based on the observed error during a previous trial (R. J. Li & Han, 2005). By iteratively applying this approach, the controller's performance can be improved. However, ILC is limited to systems that are highly repetitive, i.e., the same task is repeated many times; a constraint that the proposed method will try to circumvent.

C. Kresling Origami

The Kresling origami pattern is a folding-pattern used to create deployable cylindrical structures (also known as bellows) such as the one in Figure 2a. As shown in Figure 2b, the pattern is based on the creases that naturally occur when twist-buckling a cylindrical structure (Kresling, 2008).

Due to the strain induced on the folds when twisting or compressing the structure formed by the pattern, it forms a torsional and translational spring with nonlinear stiffness. The pattern can be defined by three parameters (Figure 3, *top-right*), depending on which the spring can be designed to be either mono- or bistable (Masana & Daqaq, 2019):



(a) Fully constructed 6-sided KOS.

(b) Twist-buckling crease pattern. Obtained from Kresling (2008).

Fig. 2: Kresling origami pattern

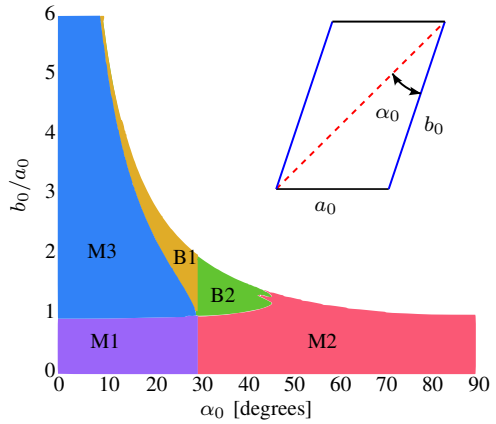


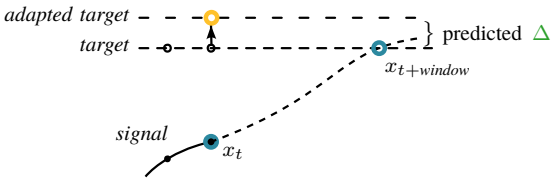
Fig. 3: Design map for a KOS with 6 sides reproduced from Masana and Daqaq (2019) and the corresponding parameters from the folding pattern, with mountain creases in blue and valley creases in dashed red.

Monostable M1, M2 and M3 type springs differ in their stable deployment height being zero, non-zero and non-zero, respectively, and their stable potential energy being non-zero, zero and non-zero, respectively. Bistable types B1 and B2 differ in their contracted stable height being non-zero and zero, respectively, and their respective potential energy being zero and non-zero. With the help of only three parameters, the map of feasible designs shown in Figure 3 can be created. Using this design map, a pattern can be designed that conforms to any of these categories. The springs used for the robot in this paper are of type B1 (Appendix D).

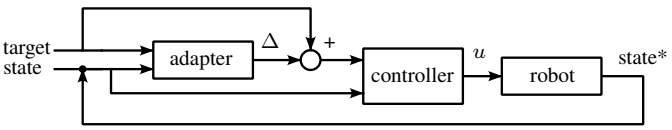
The bistable behavior of the KOS is a feature that has made the pattern popular in applications varying from mechanical switches, to aerospace structures and soft robots (Masana et al., 2024). The demonstrator robotic crawler in this paper is inspired by similar robots such as from Pagano et al. (2017) and Ze et al. (2022).

III. Methodology

The method proposed in this paper aims to provide a way of improving a given controller's performance as well as fortifying it against evolving system dynamics *without* having access to the inner workings of the controller itself. In addition, the aim is not to replace the controller entirely but to introduce an additional module to the control scheme that enables lifelong robustness to evolving system dynamics.



(a) The goal: Reach the target within the specified window. By supplying the model with the controller's target position and current state x_t, it yields a Δ that can be added to the current target.



(b) The target adaptation control scheme: The adapter is used to offset the given target before presenting it to the controller, essentially *lying* to the controller in order to improve its performance.

Fig. 4: Target adaptation

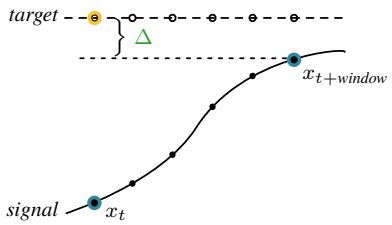
The advantage of such a method is that it could be applied to any system as a "plug-and-play" module.

The manner in which this will be achieved is by *target adaptation*. An adapter module modifies the target used by the controller to improve its performance. Consider a scenario where a controller yields a control action u to reach a certain desired target state $\mathbf{x}_d$ within a certain time window. As the system dynamics evolve, suppose this action u is now insufficient to reach the target. In response, based on the state of the system and the desired target, the adapter module "lies" to the controller about reaching $\mathbf{x}_d$ and instead provides it with $\mathbf{x}_d + \Delta$. For this adapted target, the controller now yields a larger action u^* that *does* result in the system reaching state $\mathbf{x}_d$. Figure 4a illustrates the desired outcome of the target adaptation method, the basic control scheme is presented in Figure 4b.

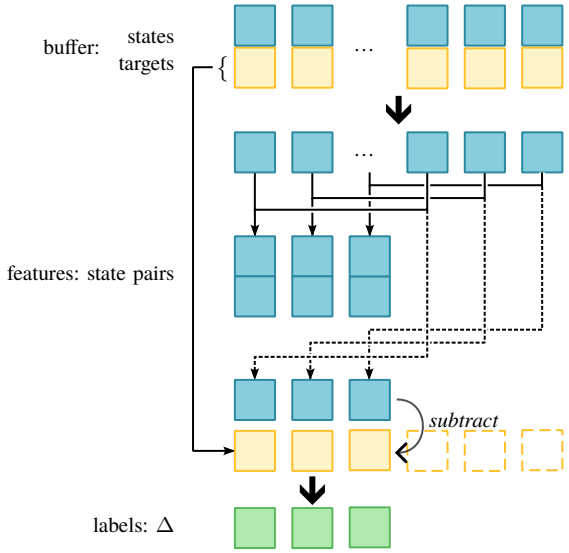
At a glance, this method resembles that of iterative learning control (ILC). However, in ILC, the adaptation to the reference signal is directly proportional to the error signal, resulting in effective behavior for a very constricted task setting, where the same task is repeated over and over. Here, the goal is to have a more universal improvement in performance, and to detach the method from the error signal.

To achieve this, the adapter is implemented as a feedforward neural network with two hidden layers with non-linear activations. To train this model to learn suitable target adaptations, interactions of the controller with the system are recorded. Specifically, a buffer stores the recorded system states with corresponding targets. The model is then provided with pairs of system states that are obtained from the buffer and are spaced by a certain time window $\mathbf{x}_t$ and $\mathbf{x}_{t+\text{window}}$. From these pairs, the model infers the target it "believes" the controller aims to reach. However, instead of learning to predict the target directly, the training labels represent the difference $\mathbf{\Delta}$ between the target $\mathbf{x}_{d,\,t}$ and the latter state in the pair (Figures 5a and 5b).

The model can be trained in a supervised manner and then deployed. During deployment, it is presented with the current state $\mathbf{x}_t$ but cannot be presented with $\mathbf{x}_{t+window}$ (since this



(a) The target adaptation problem: Based on the current position x_t and the position some prediction window later $x_{t+window}$, the model learns to predict the target the controller is trying to reach by representing it as a Δ to be added to $x_{t+window}$.



(b) The feature and label extraction process: recorded states spaced by a specified prediction window are paired up and provided as features to the adapter, which learns to predict a $\mathbf{\Delta}$, being the difference between the target $\mathbf{x}_{d,\,t}$ and state $\mathbf{x}_{t+window}$.

Fig. 5: Target adaptation during the training phase.

state is not known yet). Instead, the second state in the pair is set to the desired target. To the model, this represents a situation in which the desired target was reached from the current state within the specified time window. The idea is now that the trained model will produce a $\mathbf{\Delta}$ that–when added to the latter state in the pair (i.e., the desired target)–results in an adapted target for which the controller will produce an improved control action (Figure 4a). In short, given a current state $\mathbf{x}_t$ and a future state $\mathbf{x}_{t+window}$, the model learned to produce the desired target. Thus, when presented with $\mathbf{x}_t$ and $\mathbf{x}_{t+window} = \mathbf{x}_d$ (supposedly reached within the time window) it provides the (adapted) target that would "push" the controller toward that result.

This approach is thus completely detached from the error signal, and allows for non-repetitive settings, as opposed to ILC. ILC, however, also provides an approach of adapting the controller's action u directly. Trying to mimic this approach similarly to the target adaptation method described, one could train an adapter to predict a control action, or a series of actions, based on a state pair. The downside to this approach is that as the adapter improves, it would essentially replace the controller (during deployment, it receives the state and target, and produces an action), which is one of the outcomes to be avoided, as stated earlier. Therefore, the target adaptation was considered the more elegant approach.

Finally, the reason for predicting the difference $\mathbf{\Delta}$ between the state $\mathbf{x}_{t+window}$ and the desired target, rather than a direct prediction of the target, should also be clarified now: By designing the target adapter's output to be the difference $\mathbf{\Delta}$, the model can be initialized with parameters all close to zero (essentially leaving the desired target unchanged) and then steadily improve online as interactions of the controller and system are recorded, without the need for a pretraining phase.

IV. Illustrative Problem

To explore the proposed method, an illustrative problem is considered first. The problem consists of a simulation of a simple 1D mass-spring-damper system (Figure 6) and an imperfectly tuned PID controller. The mass is suspended in a hanging position with gravity acting on it continuously.

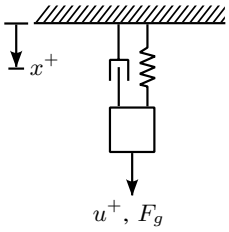


Fig. 6: Setup of the mass-spring-damper system in the illustrative problem with positive directions indicated.

The system state $\mathbf{x}$ consists of the position x and velocity $\dot{x}$. The controller aims to reach targets of the form $\mathbf{x}_d = [x_d,\, 0]$, but controls position only. At regular intervals, a random target position x_d is generated for the controller to reach and maintain. Each timestep, the adapter calculates the required Δ to be added to the intended target *position*, and the controller computes the control action based on the current position and the adapted target position. The system's response to this control input is simulated and the tuple $(\mathbf{x}_0,\, \mathbf{x}_d)$ is added to the buffer. This buffer stores a limited number of tuples corresponding to a specified window of the most recent data: Once capacity is reached, the oldest element is removed at every timestep. At regular update intervals, the adapter is trained on the data in the buffer. To ensure meaningful training, it is important that the tuples in the training data always contain the *intended* target position x_d, and not the target that was actually provided to the controller, i.e., $x_d + \Delta$. As the simulation progresses, the parameters of the dynamic system are gradually altered to simulate system degradation. The pseudo algorithm for the implementation can be found in Algorithm 1.

The simulation ran for 60s at 50Hz. The target was changed every 5s, and the adapter was trained fully online, updating every 15s. Training features were created for multiple prediction windows, this means the training features consisted of pairs $[\mathbf{x}_t, \mathbf{x}_{t+window}]$ for windows of 3, 5, 10, 15, and 25. The buffer's budget corresponded to the past 30s of data. As a baseline, a second controller and system were also simulated,

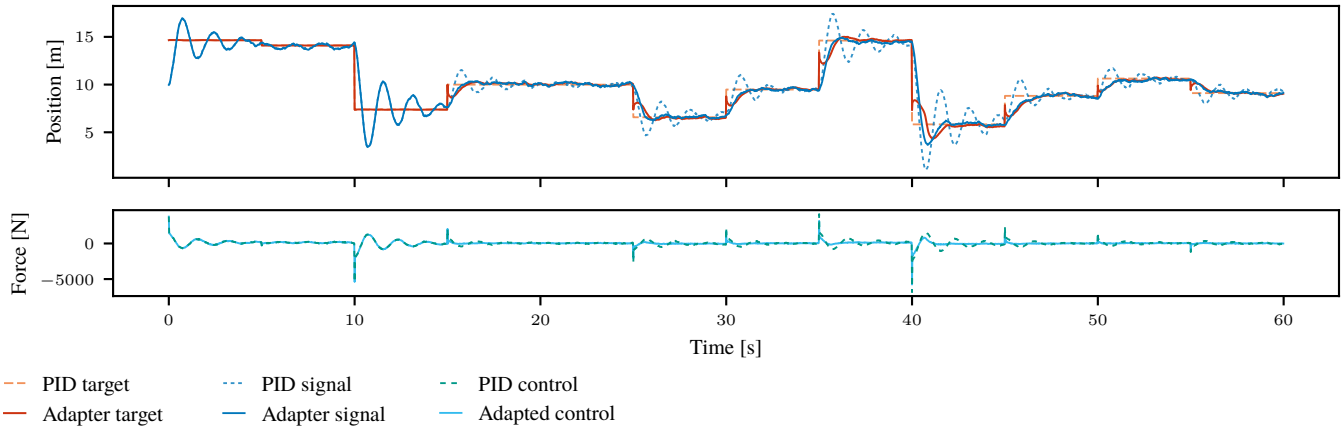


Fig. 7: The full minute simulation of the system with an aggressive controller. *Top*: The system response x and target $x_d + \Delta$ of the adapted controller, alongside the response and target of the unadapted PID controller ($\Delta = 0$). *Bottom*: The resulting adapted control inputs from the controller receiving $x_d + \Delta$ alongside the unaltered PID control inputs. During the first 15s (3 targets) the adapter has not received updates yet, and the signal is thus identical to the baseline PID signal.

Algorithm 1: Target adaptation

Data: system state $\mathbf{x}$, target $\mathbf{x}_d$, adaption Δ, control action u
Initialize model: *adapter*;
Initialize buffer: *buffer* $\leftarrow$ [];
for *each timestep* t_i **do**
  if *len(buffer) > threshold* **then**
    Remove the oldest element in the buffer;
  end
  if $t_i \mod target\_interval = 0$ **then**
    $\mathbf{x}_d \leftarrow random\_target()$;
  end
  if $t_i \mod update\_interval = 0$ **then**
    (*features, labels*) $\leftarrow$ *prep_data(buffer, window)*;
    adapter.fit(features, labels);
  end
  $\Delta \leftarrow adapter.predict([\mathbf{x}_0, \mathbf{x}_d])$;
  $u \leftarrow controller.compute\_control(\mathbf{x}_0, \mathbf{x}_d + [\Delta, 0])$;
  Simulate the system: $\mathbf{x} \leftarrow response(\mathbf{x}_0, u)$;
  Update system parameters;
  Append $(\mathbf{x}_0, \mathbf{x}_d)$ to *buffer*;
  $\mathbf{x}_0 \leftarrow \mathbf{x}$;
end

identical to the described simulation, but without the addition of an adapter. To both the baseline and adapted signal, a small amount of Gaussian noise was added to simulate measurement noise.

A. An Over-Tuned Controller

For a first simulation, a combination of a system and controller was chosen where the controller's gains were tuned too high. In Figure 7, the result of the simulation can be observed. At the start, due to the near-zero initialization of the adapter's parameters, the signal response of the system controlled by the adapted controller is indistinguishable from that of the baseline PID controller. However, as soon as the adapter has trained on the buffer, the changes to the target become visible.

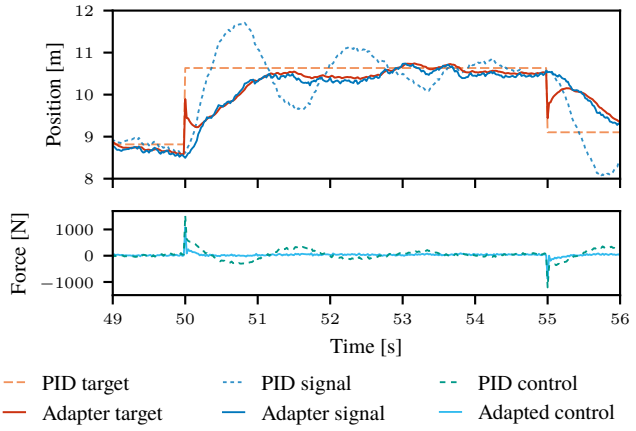


Fig. 8: A close-up of the second to last target in the one-minute simulation. The effects of the adapter can be more clearly observed: The overshoot (seen in the baseline PID signal) is countered by bringing the target closer to the current state.

The effects of the adaptation can be more clearly observed in Figure 8. Since the baseline PID controller is too aggressive, the signal overshoots and oscillates significantly before settling. In contrast, the adapter counters this by bringing the target closer to the current state. This is the logical result of the buffer containing data corresponding to the overshoots, where the adapter would often encounter training features for which the labeled target position[1] x_d is somewhere in between x_t and $x_{t+window}$. Therefore, upon receiving the input feature $[\mathbf{x}_t, \mathbf{x}_d]$ it produces and adapted target position in between x_t and x_d. As the state approaches the adapted target, the adapter starts

[1]Rather, the labeled $\Delta = x_d - x_{t+window}$, but the explanation provided is equivalent and somewhat clearer.

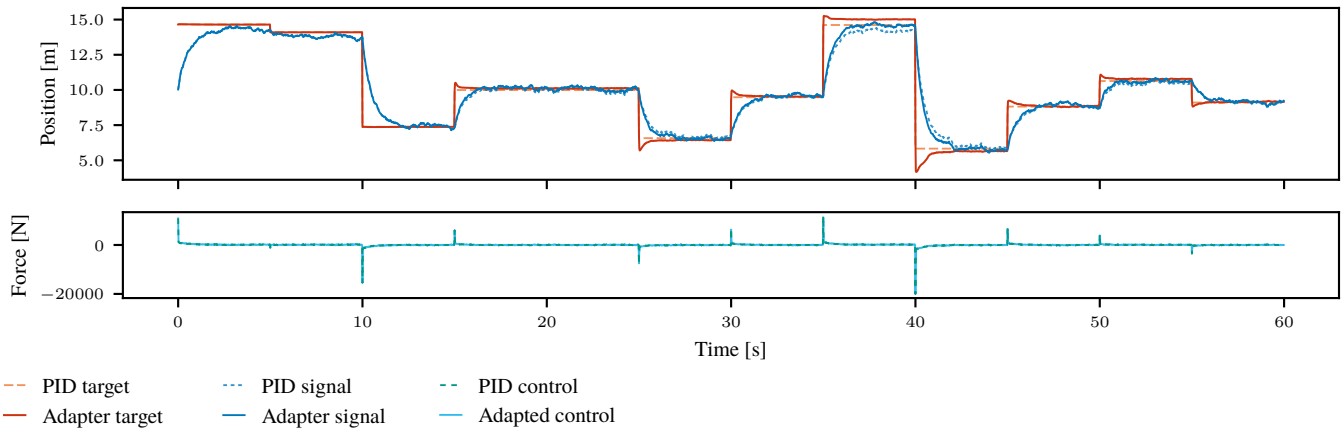


Fig. 9: The full minute simulation of the system with an under-responsive controller. *Top*: The system response and target of the adapted controller, alongside the response and target of the unadapted baseline PID controller. *Bottom*: The resulting adapted control inputs from the controller receiving the altered target alongside the unaltered baseline PID control inputs.

producing targets closer to the reference target, guiding the system towards this desired state.

B. An Under-Tuned Controller

For a second simulation, the system and controller were designed such that the controller was more conservative or *under-tuned*. Additionally, the signal produced by the baseline PID controller has a clear steady-state error. The result of the simulation can be found in Figure 9.

Once more, the adapter is able to improve the controller's performance after just a single update. In Figure 10 it can be seen that the adapted target is adjusted away from the current state. This is because training features created from the data in the buffer frequently include state pairs $\mathbf{x}_t$, $\mathbf{x}_{t+window}$ where the target position x_d is positioned further from x_t than it is from $x_{t+window}$. As a result, the adapter learns to predict adaptations that would move x_d farther from x_t, making the controller more aggressive. Another effect of the adapter is the consistent offset applied to the target: In the training features, the adapter regularly encounters pairs $\mathbf{x}_t$, $\mathbf{x}_{t+window} = \mathbf{x}_t$, which appear to represent the target state being reached, even though $x_{t+window} \neq x_d$. These features are the result of the steady-state error present in the signal, leading the adapter to learn to apply an offset to the target to counteract this.

C. Preliminary Conclusions

These qualitative results show that the adapter is able to improve controller's behavior in a simple *simulated* dynamic system. The adapter is also making target adaptations that are intuitive and logical, given the specific settings of the system and controller. The simulations thus provide a good foundation that the method is at least reasonable. For a real-world test of the method, a demonstrator robot is required (Section V) and a good test setup with a comprehensive set of metrics to quantify the results (Section VI).

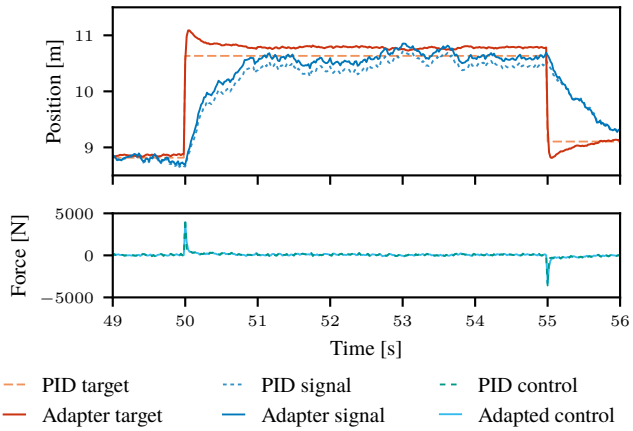


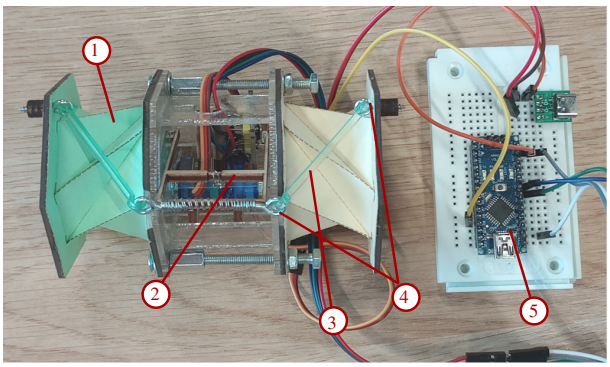
Fig. 10: A close-up of the second to last target in the one-minute simulation. The target is initially moved further from the current state to "pull" it towards the desired state in a shorter period. In addition, the adapted target is offset from the desired state as to counter the steady-state error seen in the baseline PID signal.

V. A Kresling Origami Robot

A. Mechanical Details

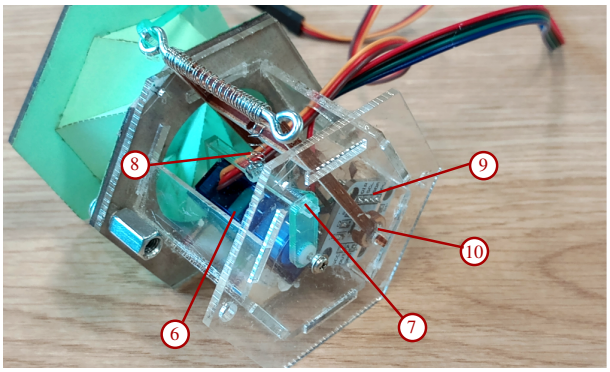
As mentioned in Section II-A, origami robots—and by extension, soft robots—are a suitable use-case for lifelong learning techniques. To evaluate the method described in Section III, a robot was designed that serves as a good analogue to the simulated problem, whilst also using Kresling origami as a key part of its functionality. Figure 11 provides important nomenclature for reference.

The robot is a simple crawler consisting of two Kresling origami springs (KOS) (Figure 11a, item 1) that are mirror images of each other, connected in series, with the actuation mechanism in between. The KOSs in question are of the bistable B1 type (Section II-C) with perforated folds to facilitate actuation (refer to Appendix D for design details



1) KOS
2) Actuation arm
3) Connecting link
4) Eye screw
5) Arduino

(a) The Kresling origami robot.



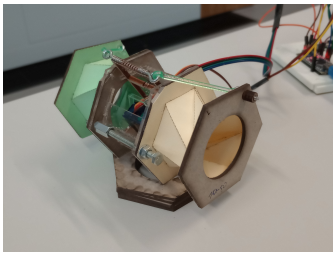
6) Servo motor
7) Servo arm
8) Metal spring
9) Angle sensor
10) Magnet

(b) Actuation mechanism

Fig. 11: Annotated views of the robot and its most important actuation mechanisms.

and templates). Both springs can expand and contract in sync, producing a crawling-like motion when combined with specially designed "feet" (see Figure 1 in Section I). To demonstrate target adaptation, however, the movement of the robot as a whole is not very important. Rather, the state of the KOSs is the state of interest that will be controlled.

A KOS can be actuated both by linear motion (pushing or pulling on the spring) and by rotational motion (twisting). Since the two motions are directly related (i.e. twist-buckling, discussed in Section II-C), a contracting or expanding motion can be measured as the angle of the KOS's twist. The twist is induced by turning a rotating arm (2) which is connected to the ends of the KOSs by two links (3) that hook into eye-screws (4). To turn the arm, a micro servo motor (6) is employed. In the illustrative problem, the controller yields a force to exert on the system in order to reach the target position. However, most micro servo's come with a near perfect position controller built-in, which cannot be accessed or changed. Therefore, to bridge the gap between the simulation and the real-world implementation, the position output of the servo is converted to



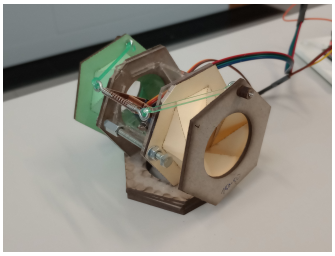


Fig. 12: *Left*: The robot at its fully extended state, corresponding to the "home" position of the servo and actuation arm. *Right*: The fully contracted robot, corresponding to the maximum (capped) servo output, near the mechanical limit of the actuation arm.

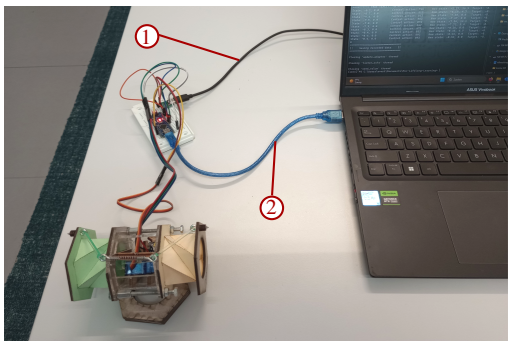
a force output by the use of a small, metal spring (8). The servo then does not directly act on the actuation arm, but rotates a servo arm (7) instead, which pulls the spring and thereby exerts a force on the actuation arm. As previously mentioned, the state of the KOS can be measured as the angle of twist, since it is directly related to its contraction. Likewise, assuming perfectly rigid links with negligible clearance between the hooks and eye-screws, the angle of twist is directly related to the angle of the actuation arm. In the full implementation, the robot's state $\mathbf{x}_t$ will thus be measured as the actuation arm's angular position and velocity $[\theta, \dot{\theta}]$. This angle is measured using a contactless sensor (9) that can measure the on-axis rotation of a small magnet (10) attached to the actuation arm. The sensor relays this information to an Arduino Nano (5) board, which runs a PID controller that returns the control action, i.e., an angular position for the servo. Furthermore, there are mechanical limits for the rotation of both the actuation arm and servo arm, imposed by the edge of the container housing the actuation mechanism and the supports holding the servo motor, respectively (Figure 12, *right*). However, the design is such that the servo's mechanical limit corresponds to a θ below the actuation arm's limit.

B. Software Implementation

The software implementation consists of two main components: The nominal control of the robot, and the target adaptation.

The Arduino board is responsible for the nominal control, implemented as a simple PID controller. Additionally, it takes care of calibrating the measurements upon startup, i.e. (re)setting the actuation and servo arm to their minimum "home" position (Figure 12, *left*).

Target adaptation is implemented on an external computer. Through a serial connection (Figure 13, item 2) with the Arduino, it both receives the measurements to be stored in the buffer, and sends the adapted targets to be used by the PID controller. The main process was summarized in Section IV by Algorithm 1. However, on the computer, the several steps are divided amongst separate modules that act in parallel so that real-time performance is ensured. A more detailed description of the software implementation can be found in Appendix A.



1) Servo motor power supply
2) Serial connection and Arduino power supply

Fig. 13: The test setup.

VI. Experimental Setup

For each experiment, a minimum of two sets of KOSs was required: One set serving as a baseline, with the Arduino receiving unadapted targets, and the second set used for testing target adaptation. All experiments used a 60s buffer and 15s intervals between adapter updates, with the first update at 60s. The adapter trained on prediction windows of 3, 5, 10, and 15 steps. Both runs of each experiment use the same preset sequence of random targets. This sequence amounts to 5min of deployment time and is repeated according to the full experiment duration. The signals corresponding to each 5min sequence are recorded and saved for evaluation. This way, the evolution of the performance metrics can be studied, and a fair comparison between the baseline PID controller and adapted controller is ensured. Figure 13 shows the general experimental setup.

A. Sign Conventions and Unit Clarification

Before the experiments and results can be properly understood, it is necessary to clarify certain sign and unit conventions.

For example, the contactless magnetic sensor denotes a clockwise rotation (as viewed from the top) w.r.t. the datum as a positive angle, and likewise, a counterclockwise rotation as a decrement in the angle (Rob Tillaart, 2024). In practice, this means that a positive control input from the servo (i.e. a positive angle) results in a decrement in θ. Because the actuation arm's calibration point is set when the servo is at its minimum output, the operational range of the actuation arm is entirely in the negative domain. Figure 14 illustrates the servo arm and actuation arm in a configuration representative of a typical scenario during the experiments, with the signs of the angles indicated.

In addition, The *Servo* Arduino library uses the `writeMicroseconds(value)` function to write to the servo. `writeMicroseconds(value)` maps a servo's range to a value between 1000 and 2000µs (depending on the servo's manufacturer), relating to the pulse width in microseconds of the signal that the Arduino sends to the servo motor (Arduino, 2024).

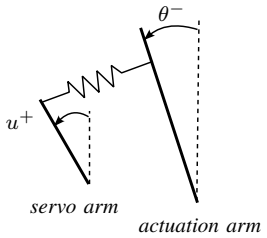


Fig. 14: Nominal signs of the angles for the servo and actuation arm during operation.

Finally, during the experiments, the Arduino initialized a *count* value to include in the tuple sent to the computer. This counter starts at 0 and simply increments by 1 with each iteration of the Arduino's loop. Since the interactions between the Arduino and computer and the several threads on the computer are independent of timing (i.e., the modules do not have to "wait for each other"), the counter's value is essential to accurately plot the several signals recorded during the experiments.

B. Performance Metrics

The performance metrics used to evaluate the 5min sequences are the mean rise time (MRT), mean settling time (MST) of the system response, mean overshoot (MO) of the system response, mean absolute error (MAE) of the system response, mean absolute velocity (MAV) of the system response, and normalized total variance (NTV) of the control signal, which are common metrics for evaluating controller performance (Alfaro & Vilanova, 2016; Marshall, 1978).

The mean rise time is calculated as the mean of all the rise times corresponding to each target in the 5min sequence. The rise time is defined as the time it takes the robot to come within 10% of the target state, as calculated from the initial position.

Similarly, the settling time is the average of the settling times for each target. The robot state is considered settled when it remains within 5% of the target state, as calculated from the initial position.

The mean overshoot is the average of the maximum overshoots observed for each target.

The mean absolute error represents the average difference between the robot's angle θ and the true target θ_d.

The mean absolute velocity refers to the measurements of $\dot{\theta}$. This metric is meant to provide a measure for the "chaoticness" of the robot's signal response.

The normalized total variance is calculated as the sum of the absolute of all increments and decrements in the control action, normalized over time[2]. This aims to quantify the controller's effort.

VII. Experiments and Results

The performance of the target adapter is evaluated over three types of experiments. In the first type, the PID controller's

[2]Actually, it is normalized over the *count*, see Section VI-A.

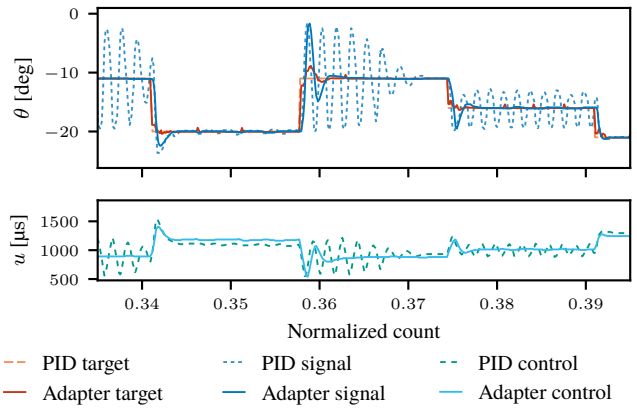


Fig. 15: Qualitative comparison between the behavior of the baseline controller and target adapted controller for the *slow-aggressive* PID controller.

TABLE I: Percentual change after applying the target adapter for the *slow-aggressive* PID experiment.

MRT	MST	MO	MAE	MAV	NTV
+67%	-30%	-33%	-60%	-73%	-70%

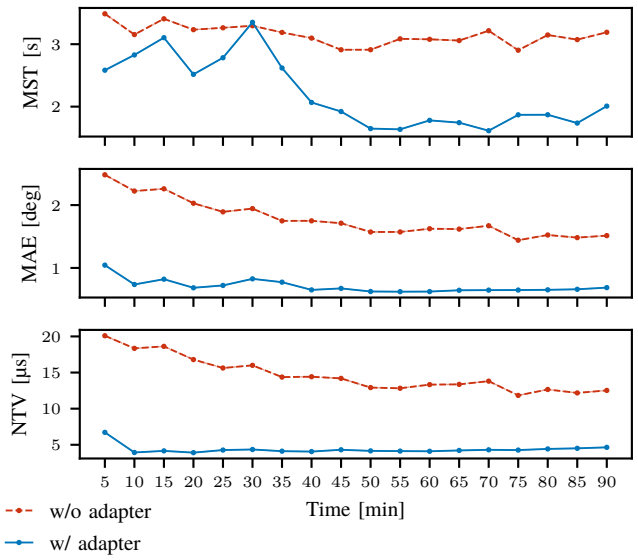


Fig. 16: Metrics evolution over the duration of the experiment for the *slow-aggressive* PID controller. Each point represents the calculated value of the metric on identical 5min sequences of targets. Plots for all metrics and numerical values can be found in Figure C.1 and Table C.I, respectively.

tuning is varied, while maintaining consistent KOS characteristics (Section VII-A). For the second group of experiments, the system dynamics are varied, namely the degree of perforation in the folds of the KOS, while keeping the PID controller identical across tests (Section VII-B). Finally, the performance is tested for a crawling gait to observe the effect of the method in a highly restricted setting (Section VII-C). For the comprehensive set of results, refer to Appendix C.

A. Varying the PID Controller

For the first experiment, the method's robustness to the use of different PID controllers is tested. The robot is equipped with the standard set of KOSs, using perforations of 1.5–0.5mm cut–skip ratio. During this experiment, the values of the targets in the random sequence are integer values ranging between -3 and -25 degrees. The robot has 5s to reach and settle around a target before it is provided with the next target in the sequence.

1) A controller that is slow but aggressive: The first PID controller has a disproportionally high integral gain, resulting in an aggressive but slow to respond controller. This controller yields particularly interesting results in combination with the bistable nature of the KOS. Figure 15 shows the comparison between the signal responses from the baseline controller and adapted controller, illustrating the behavior of the controller: The adapted target is brought closer to the system state. The effect of this can be seen in the control signal, displaying less oscillatory behavior and (slightly) lower peaks. Contrary to the baseline controller's state response, which oscillates wildly, the adapted controller overshoots once or twice and then settles.

During the full 90min duration of the experiment, each 5min sequence was recorded and evaluated using the metrics discussed in Section VI-B. Table I shows the results of implementing the target adapter as the percentual change in the average values of the metrics compared to the baseline controller.

The rise time is the only metric that increases after applying the adapter, all other metrics are consistently lower from the start. This is demonstrated in Figure 16, most notably for the mean absolute error and total variance. The settling time also improves immediately, but only decreases significantly after approximately half the duration of the experiment.

Interpretation: The conjecture is that the potential barrier between the two stable states of the KOS slows down the controller's response, resulting in a higher control input. As the barrier is overcome, the KOS now complements the controller's action, resulting in the overshoot. Depending on the vicinity of the target to the KOS's potential barrier, the controller can enter an unstable state. As the potential barrier to transition between the two stable states of the KOS decreases (due to deterioration), the baseline controller has less difficulty maintaining a stable response. This explains the baseline's *slight* decrease in settling time, and noticeable decrease in mean absolute error and total variance. The target adaptations facilitate a smoother transition to the desired state with higher rise time but reduced overshoot, thereby minimizing the risk of the controller entering an unstable cycle. From Figure 15 it seems that this mechanism is most effective after the first overshoot.

2) An under-tuned controller: The second PID controller is tuned with low gains, resulting in slow and under-responsive behavior. A visual comparison between the baseline and target adapter can be found in Figure 17. The targets for the adapter controller are shifted to more extreme positions, and then gradually move to the intended target. The resulting response

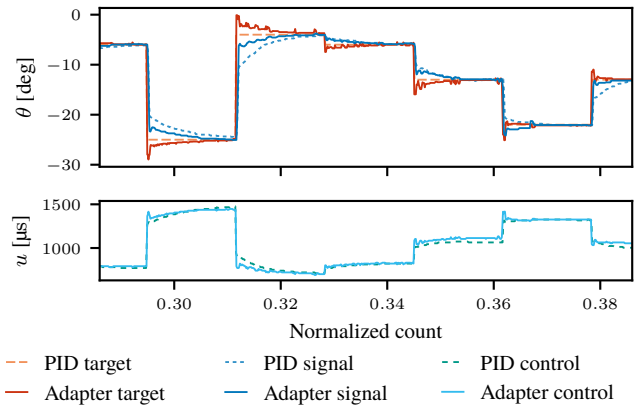


Fig. 17: Qualitative comparison between the behavior of the baseline controller and target adapted controller for the *under-tuned* PID controller.

TABLE II: Percentual change after applying the target adapter for the *under-tuned* PID experiment.

MRT	MST	MO	MAE	MAV	NTV
-44%	-5%	+614%	-44%	+13%	+99%

is closer to the intended target, and is there faster than the baseline controller's response.

The resulting changes in metrics after applying the adapter are presented in Table II. The evolution of the mean rise time, overshoot, and total variance is shown in Figure 18. The controller with adapter clearly has lower rise times and absolute errors than the baseline, but also exhibits an increase in overshoots. It should be noted that the large relative increase in overshoot stems from the baseline controller having close to zero or even slightly negative overshoot. The adapted controller results in consistently higher total variance, rising further as the experiment progresses. A similar steady increase can be observed for the rise time and absolute error of the baseline controller.

Interpretation: The signal in Figure 17 shows a clear reason for the lower rise times, namely that the target is shifted to more extreme positions as to yield larger control inputs from the controller. The adapter makes the controller more aggressive, as seen in the lower rise times, larger overshoot, higher mean absolute velocity, and increased controller *effort* (total variance). The adapter also seems to introduce unnecessary adaptations once the steady state is reached, further increasing the mean absolute velocity. These small oscillations are insignificant compared to the overall improvement in the mean absolute error resulting from the lower rise time. Finally, the steadily increasing rise time and mean absolute error of the unadapted controller going hand in hand with a similar steady increase of adapted controller effort might suggest that the system dynamics during both experiment runs are evolving slightly, but with the adapter compensating for it.

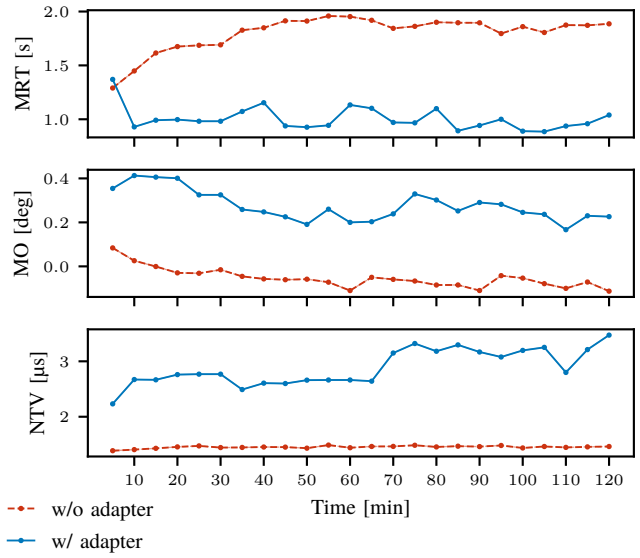


Fig. 18: Metrics evolution over the duration of the experiment for the *under-tuned* PID controller. Each point represents the calculated value of the metric on identical 5min sequences of targets. Plots for all metrics and numerical values can be found in Figure C.2 and Table C.II, respectively.

TABLE III: Average percentual change after applying the target adapter for the 0.25–0.25mm and 0.5–0.5mm dynamics experiment.

MRT	MST	MO	MAE	MAV	NTV
-64%	-36%	+323%	-37%	+8%	+13%

B. *Varying the System Dynamics*

For the second type of experiment, the PID controller is tuned to a highly responsive performance on a robot with KOS perforations of a 1.5–0.5 mm skip-to-cut ratio. The KOS perforations used are 0.25–0.25 mm, 0.5–0.5 mm, 1.5–0.5 mm, and 2–0.5 mm, representing various stages of KOS degradation. Once again, the values of the targets in the random sequence are integer values ranging between -3 and -25 degrees, and the robot has 5s to reach and settle around a target.

1) 0.25–0.25 & 0.5–0.5 Perforations: The 0.25–0.25mm and 0.5–0.5mm perforations represent KOSs in a state of lower degradation than the standard 1.5–0.5mm perforation. The results of applying the target adapter to these settings are averaged for both experiments and are presented in Table III. The evolution of the mean rise time, overshoot, and total variance over the course of the 1h experiment is shown in Figure 19, showing the average of the results from both experiments. The metrics display similar trends to those observed for the *under-tuned* PID controller: On average, the adapted controller has lower rise times, settling times, and absolute errors, but higher overshoot, absolute velocity, and total variance. Another observation is that the difference in the baseline controller's performance between the two experiments drifts further apart over the full duration. However, the adapted controller's performance over the two experiments appears less

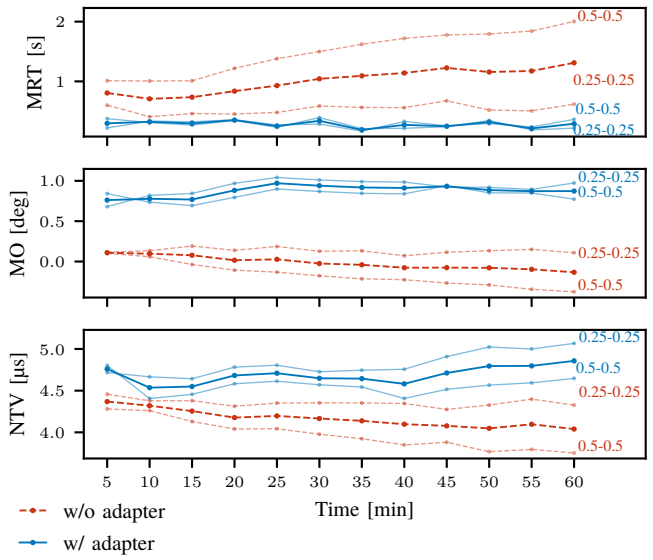


Fig. 19: Metrics evolution over the 1h duration of the experiment for KOSs in a lower state of degradation (0.25–0.25mm and 0.5–0.5mm perforations, see annotations), with their average overlayed. Each point represents the calculated value of the metric on identical 5min sequences of targets. Plots for all metrics and numerical values can be found in Figure C.3 and Table C.III, respectively.

varied.

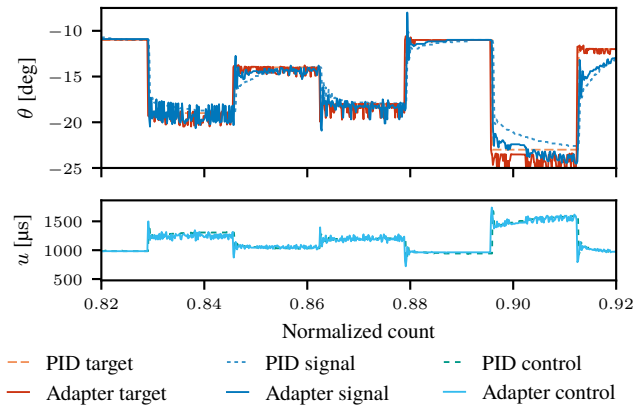
Interpretation: Since the perforations aim to represent a less degraded KOS, i.e. a stiffer KOS, the gains of the PID (tuned to a 1.5–0.5mm perforated KOS) are likely set too low. The situation is thus analogous to the *under-tuned* PID experiment, and this shows in the behavior of the target adapter: The targets are shifted toward more extreme positions, resulting in the lower rise time. The observation that the metrics for the baseline controller drift further apart between the two experiments is a good sanity check that the dynamics of the springs are indeed different. The fact that the target adapter's metrics are less varied is then a good sign that the adapter is able to generalize and bring both systems to a stable performance.

2) 1.5–0.5 & 2–0.5 Perforations: The 2–0.5mm perforation aims to resemble a KOS that has been more deteriorated. The PID controller is initially tuned for the 1.5–0.5mm KOS. For the 1.5–0.5mm KOS experiment, with the runtime set to 2h, the aim is to assess how the controllers perform as the system gradually shifts into a more over-tuned condition.

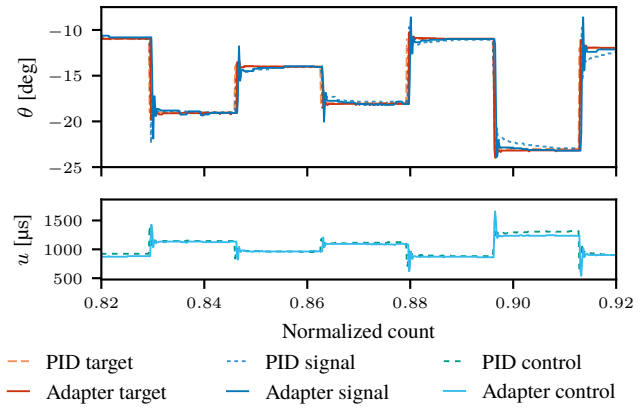
Inspecting Figure 20a, it can be observed that the signal of the adapter rises toward the intended target faster than the baseline signal. Overshoots are introduced by the adapter as well, but what draws the most attention is the apparent noise present in the signal. By the end of the experiment, the target adaptations have become seemingly random, resulting in a chaotic control signal and system response.

Like for the 0.25–0.25mm and 0.5–0.5mm experiment, the metrics presented are averaged over the two settings and can be found in Table IV. These results show that the general performance of the target adapter consists of higher settling times, overshoot, mean absolute velocity and total variance compared to the baseline, especially as the experiments progress. In fact, the only metric that has a significantly lower value for the target adapter than for the baseline, is the rise time. Figure 21 shows that the decrease in mean absolute error is only present for the 2–0.5mm perforated KOSs. However, also noteworthy in Figure 21 is that the adapted controller initially demonstrates better performance concerning settling time, overshoot and mean absolute error. Indeed, Figure 20b illustrates that, in the first 30min of the experiment, the target adapter produces minimal and relatively stable adaptations, leading to improved rise times and lower absolute error.

Interpretation: In the case of the heavily deteriorated KOSs, the target adaptation seems to degrade the performance.



(a) A comparison late in the experiment (last 5min sequence).



(b) A comparison early in the experiment (5min sequence that ends at minute 30).

Fig. 20: Qualitative comparison of the signal response for the 2–0.5mm perforation. Also showing a comparison between the signals of the target adapted controller, late vs. early into the experiment.

TABLE IV: Average percentual change after applying the target adapter for the 1.5–0.5mm and 2–0.5mm dynamics experiment.

MRT	MST	MO	MAE	MAV	NTV
-43%	+20%	+53%	-5%	+34%	+74%

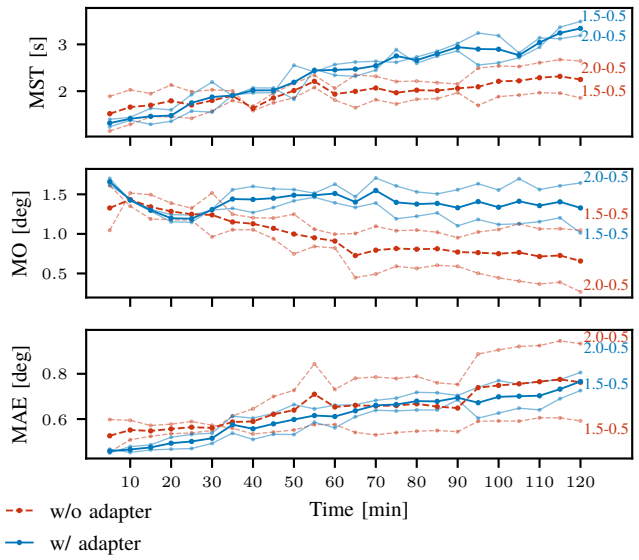


Fig. 21: Metrics evolution over the 2h duration of the experiment for KOSs in a higher state of degradation (1.5–0.5mm and 2–0.5mm perforations, see annotations), with their average overlayed. Each point represents the calculated value of the metric on identical 5min sequences of targets. Plots for all metrics and numerical values can be found in Figure C.4 and Table C.IV, respectively.

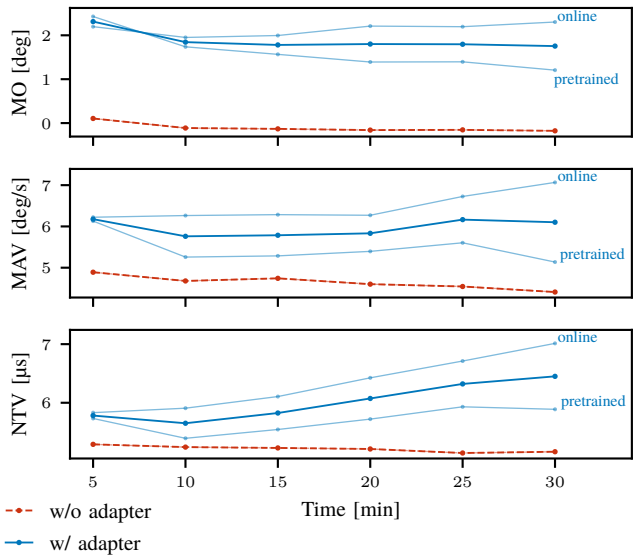


Fig. 22: Metrics evolution over the 20min duration of the crawling-gait experiment, with the average of both adapted controllers overlayed. Each point represents the calculated value of the metric on identical 5min sequences of targets, in this case alternating between values near to -18 and -8 degrees, respectively. Plots for all metrics and numerical values can be found in Figure C.5 and Table C.V, respectively.

The improved rise times can be attributed to the highly oscillatory nature of the signal response, rather than genuine performance gains. However, the fact that the target adapter shows improvement over the baseline controller in the early stages of deterioration is promising. The highly responsive PID controller is quick to react, reducing the need for extreme adaptations like those seen in the previous experiments. As the KOS further degrades, it sometimes enters an unstable cycle: much alike the oscillations in Figure 15, Section VII-A, but at a higher frequency. This results in training data that contains many contradicting features, which results in increased noise in the adapter's predictions. Noisy targets in turn produce noisy control actions and consequently noisy signal responses.

C. Crawling Gait

The final experiment emulates the mode of operation that the robotic crawler was designed for. The targets in the sequences alternate between two angles on opposite sides of the potential barrier, specifically -8 and -18 degrees, mimicking the contraction and expansion typical of a crawling motion. To introduce variability and avoid identical datapoints, a small normally distributed random offset is added to each alternating target. The goal of this experiment is to evaluate how a constrained operational setting affects the adapter's compensation for the overshooting effect caused by the potential barrier. The experiment uses a KOS with a perforation 1.5–0.5mm and the highly responsive PID, tuned to this perforation. Given the rapid degradation of the potential barrier in the early stages, the experiment only runs for 20min. A baseline controller is compared with two target adapter configurations: one pretrained and then fine-tuned online, and one that is entirely trained online.

TABLE V: Average percentual change after applying the target adapter for the crawling gait experiment.

MRT	MST	MO	MAE	MAV	NTV
-91%	-72%	+1890%	-54%	+29%	+15%

The effect of the target adapter is given by Table V. The results show consistently lower rise and settling times, and mean absolute error for both implementations of the target adapter compared to the baseline controller. These improvements come at the cost of significantly higher overshoot. The difference in metrics for the pretrained and online trained implementations appears negligible, except for the mean overshoot, mean absolute velocity, and normalized total variance, as shown by Figure 22.

Signal responses are shown in Figure 23, note that the adapted signal is only shown for the fully online trained adapter, but it is very similar to that of the pretrained adapter. An improvement in rise time is clearly observable, and shows that the improved settling time directly results from this, rather than the baseline controller having an unstable signal. In fact, the signal from the baseline controller resembles that of an under-tuned controller. Target adaptations appear to be minimal, being only visual in close-ups of the signal. This corresponds to the relatively low difference in total variance between baseline and adapted controllers, observed in the metric evolution.

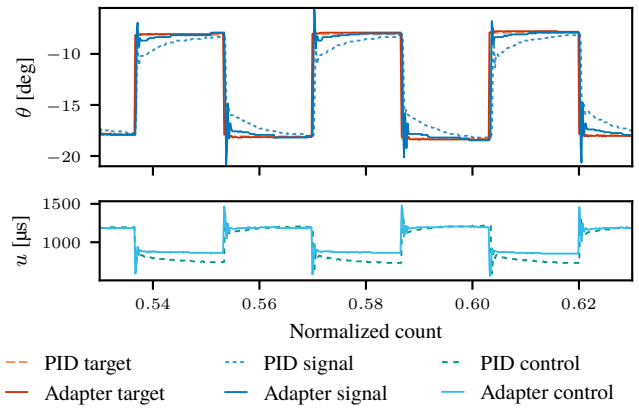


Fig. 23: Qualitative comparison between the behavior of the baseline controller and target adapted controller for the crawling gait experiment.

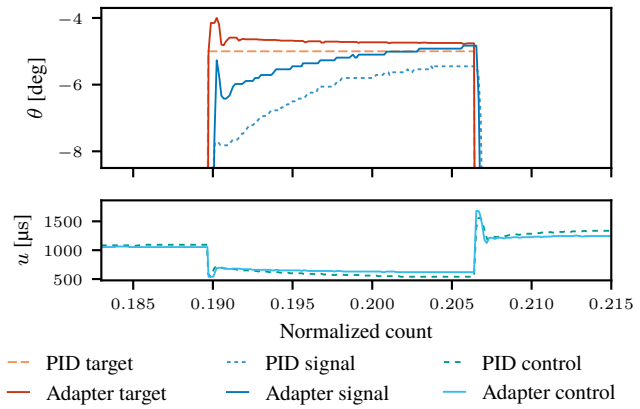


Fig. 24: A close-up from the 0.5–0.5mm perforation with target adapter. The signal response generally resembles that of Figure 10 from the simulated problem.

Interpretation: Although the adaptations made to the targets appear negligible for both the pretrained as the online trained adapter, the metrics and signal response show a definite improvement. However, the cause of how these minor adaptations result in the observed improvements is not immediately clear: the baseline controller shows signs of being under-tuned, but the adapter does not produce adaptations similar to those of an under-tuned controller in Section VII-A. On the other hand, the consistent improvements seen across both adapter configurations challenge the notion that these improvements are merely coincidental.

VIII. Discussion

Section VII already provided some short interpretation and discussion on the various results presented there. This section will provide insight in the general trends observed and discuss the limitations of the method that were encountered.

A. Key Insights

A general trend in the effect of the target adapter can be observed: Whenever a controller-system configuration exhibits under-responsive characteristics, such as the under-tuned controller in Section VII-A and the 0.25–0.25mm & 0.5–0.5mm perforations in Section VII-B, the target adapter induces more aggressive controller behavior. Likewise, when the controller is too aggressive from the start, target adaptions tend to moderate or constrain the behavior.

This follows from the interpretation of the evaluation metrics in these scenarios. For example, under-responsive systems exhibit high rise times, and therefore high settling times. These systems are also characterized by low overshoot, low mean absolute velocity, and low controller *effort* (total variance). When the adapter produces more extreme target positions, rise time and settling time decrease, but the overshoot increases. The controller is thus "making an extra effort", which is observed in the higher total variance: initially setting a more extreme target position, then gradually bringing it closer to the desired target, etc. The overall faster response results in the increased

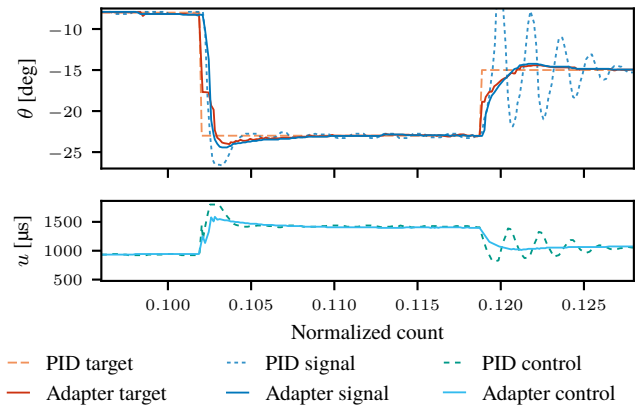


Fig. 25: A close-up from the *slow-aggressive* PID controller with target adapter. The manner in which the target is adapted resembles that of Figure 8 from the simulated problem.

mean absolute velocity. As can be observed in Figure 24, this behavior resembles that of the under-responsive simulation in the illustrative problem (Section IV-B) very closely.

Similarly, systems with an over-aggressive controller, i.e. the *slow-aggressive* controller experiment and the 2–0.5mm & 1.5–0.5mm perforations, exhibit low rise times, but high settling times: the high gains "push" the state toward the desired target quickly, and then overcompensate for the high overshoot, resulting in an unstable response near the target. This explains the high total variance. The adapter moves the target to less extreme positions, compensating for the overshoot in advance. This prevents the system from entering an unstable feedback loop by preventing the control inputs from getting overly aggressive in the first place. Thus, even though target adaptions are made, the overall controller effort is decreased. Figure 25 provides a clearer view of the process. Again, a resemblance to the equivalent scenario in Section IV-A can be observed. The contradicting results from the 2–0.5 & 1.5–0.5mm perforations, see Figure 21, will be addressed in Section VIII-B. However, as mentioned in Section VII-B, the adapted controller *does* outperform the baseline in the early stages, and it is during this

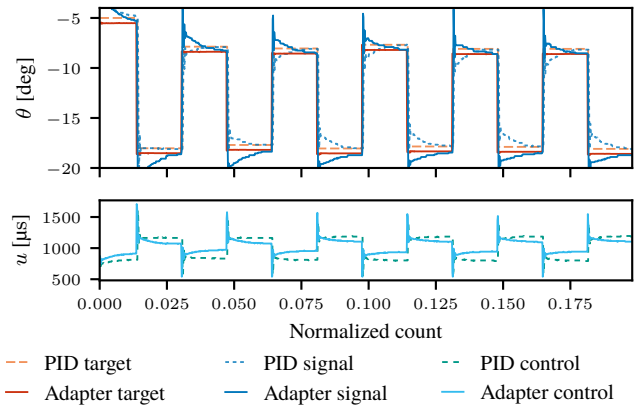


Fig. 26: Minute 1 of the pretrained adapter for the crawling gait experiment. Adaptations are immediately added to the target, resulting in worse behavior.

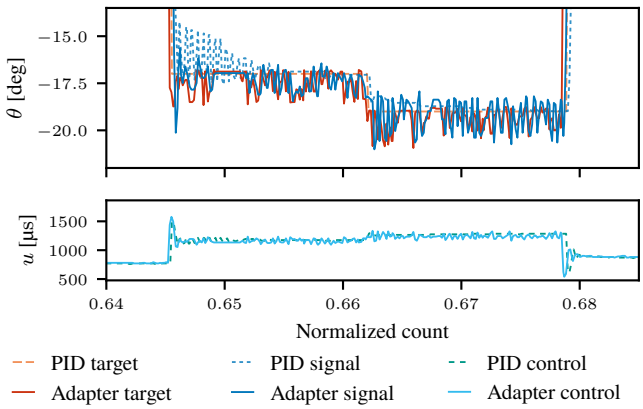


Fig. 27: A close-up of the 2–0.5mm experiment. The oscillations in the signal response are recorded at an insufficient frequency to contain any meaningful relationship between spaced-out state pairs and the target. As a result, the adapter's output becomes more chaotic as well, further polluting the training data.

phase that the same trends described earlier can be observed.

These two observed trends in the target adapter's effect suggest that the method is inherently self-correcting. By constantly retraining the adapter on features derived from its own influence on the signal response, an under-responsive adapted controller will become more aggressive, up to the point that the adapter needs to moderate its behavior again. This idea is supported by the observation that the evolution of the metrics for adapted controllers tends to stagnate (most clearly visible in Figures 16, 18 and 19). Supposedly, the adapted controller finds a good compromise between moderate and aggressive behavior, and is able to maintain a consistent performance from that point onward.

Another insight is provided by the difference between the pretrained and online trained adapter in the crawling gait experiment. In Section VII-C, Figure 22, the metrics of the pretrained and online implementation drift apart as the experiment progresses. However, the qualitative results display very little difference between both implementations, except at the very start of the experiment: The pretrained adapter immediately adapts the target, but in a counter-productive manner, as seen in Figure 26, whereas the online trained adapter first makes no adaptions at all.

The pretrained implementation is able to recover within one or two updates. Nonetheless, this counter-productive behavior, and the need for pretraining data makes a fully online implemented adapter the more attractive solution.

B. Limitations

As discussed in Section VII-B, the experiment using 1.5–0.5 & 2–0.5mm perforations in combination with a highly responsive PID controller yielded better results for the baseline controller than for the adapted controller. The hypothesis is that the training data obtained from the signal response gradually contains more conflicting features, as illustrated by Figure 27. As the controller is highly responsive, the frequency at which the signal response is stored in the buffer is too low to capture sufficient detail.

However, this high responsiveness alone is not solely responsible for the adapter's deteriorating performance, since the adapter initially manages to slightly improve performance. The same baseline controller is used in the 0.25–0.25 & 0.5–0.5mm perforation experiments, as well as in the crawling gait experiments (Sections VII-B and VII-C, respectively). It seems to be the specific combination of a highly responsive controller–or conversely, a low sampling rate for the buffer–with a more deteriorated KOS that leads to the issue. Once the system enters an unstable feedback loop, the oscillations occur so frequently that any meaningful relationship between sequential datapoints is lost, an observation which becomes clear by comparing Figures 25 and 27. As a result, training features obtained from near-identical regions in the signal response can vary drastically. The adapter thus starts producing less meaningful adaptations, entering a vicious cycle.

This explains why the combination of 1.5–0.5mm perforations and a *slow-aggressive* controller poses no problem, nor does the same perforation combined with the highly responsive controller during the first 30min (also demonstrated by the crawling gait experiment). Similarly, the 0.25–0.25 and 0.5–0.5mm perforations effectively prevent the controller from entering an unstable state.

C. Validity of the Results

Several important aspects regarding the validity of the results should be addressed.

1) Reproducibility of Identical KOS: Since a key part of understanding the results is the use of a PID controller as a baseline, it is important to confirm that the comparisons are representative. Section VI already stressed that identical target sequences were used for each comparison. The focus thus shifts to the possible differences in the characteristics of supposedly identical KOSs.

The KOS templates were all designed with identical parameters, exempting the perforations, and were cut out using a

laser-cutter. They were, however, assembled by hand. Several steps were taken to mitigate or minimize variability introduced during the assembly.

First, the required KOSs for a certain experiment setting–usually four, since two are required for the reference run, and two for the adapter test–were all made in a single batch, following a preset routine. Additionally, all KOSs were then conditioned by exactly one cycle of contracting and expanding it. This procedure reduces potential differences in the cutting and assembly process.

Second, the nature of the robot's design requires two KOSs per experiment run: as described in Section V-A, the state of the robot is represented by the angle of the actuation arm. This arm is restrained on both sides by a KOS, each a mirror image of the other. In practice, the combined contribution of both springs averages out, and the influence of any single KOS is minimized.

Finally, during each run of an experiment setting, a "reference minute" was recorded with intervals of 30min. During these reference minutes, and regardless of the specific experiment setting (i.e., with or without the adapter), the robot ran the relevant reference controller on a 1min target sequence. This meant pausing the experiment, disabling the adapter, and evaluating the performance metrics on a preset 1min sequence. These minutes were evaluated using the same metrics as the experiments to compare the similarity of the KOSs involved. Comparisons of these metrics are presented in Appendix B, but it is sufficient here to note that variations in metric values for reference minutes are insignificant compared to the difference introduced by the implementation of the target adapter. These measures allow for valid interpretation of the results.

2) Uninterpreted Results: For certain experiments, the nature of the adapter's influence causing the improvement remains unclear. Qualitative results for both adapter configurations of the crawling gait experiment, and the early stage of the 1.5–0.5 & 2–0.5mm perforation experiment (Figures 20 and 23) do not indicate significant adaptations to the target. This might lead to disregarding the results as coincidences. However, the improvement is consistent over all these experiment settings, with notable similarity in the reference signal responses for all settings, as well as adapter signal responses (i.e., signals from seemingly slightly under-responsive systems, and highly responsive systems, respectively).

IX. Conclusion

The goal of the target adaptation approach is to provide a simplified and intuitive alternative to more opaque control techniques commonly seen in soft-robotics applications. The idea is that a simple or traditional method can be used to handle the base control of the robot, and that a separate module improves the performance of that controller. It achieves this by providing the controller with adapted targets, but without altering the controller's parameters.

In simulation, it was shown that such an approach offers near instantaneous improvement for the control of a simple mass-spring damper system.

The tests on a demonstrator robot, with a design relying on Kresling origami springs, show that the method is robust to varying controller tuning and varying system dynamics. The target adapter module tends to make passive controllers more aggressive, and vice versa, resulting in a consistent performance. Usage of the method on a constrained operational domain shows that both pretrained and online trained adapters manage to improve the performance, with a preference for the latter.

A limitation is presented in the form of the sampling frequency at which the signal response is stored for training the module: The frequency has to be high enough to capture enough detail. If the sampling frequency is too low *and* the recorded data contains some unstable oscillations, the adapter is unable to find meaningful relationships between the state pairs and targets it uses for updating itself. Updates on such meaningless features lead to a negative feedback loop for the adapter's performance, which it is unable to escape.

Possible solutions to the problem are methods that allow the adapter to either recall previous settings or maintain a more stable performance. An implementation of some kind of memory or knowledge base, or more envolved updating techniques like *learning without forgetting* (Z. Li & Hoiem, 2018), could potentially allow the adapter to escape or avoid such vicious cycles. Additionally, since the adapter seems to produce unnecessary adaptations even when the intended target has been reached, a mechanism that forcefully removes or limits adaptations once that target is reached could help reduce the chances of becoming unstable.

A sensitivity analysis on the effect of the various parameters involved in the method could potentially isolate the issue. Examples are the (combination of) prediction windows used, the sampling rate of the buffer, the frequency of training updates, and the contents of the states in training features.

References

Alfaro, V. M., & Vilanova, R. (2016). Control System Evaluation Metrics. In *Model-reference robust tuning of pid controllers* (pp. 21–28). Springer International Publishing. https://doi.org/10.1007/978-3-319-28213-8_3

Annaswamy, A. M., & Fradkov, A. L. (2021). A Historical Perspective of Adaptive Control and Learning. *Annual Reviews in Control*, *52*, 18–41. https://doi.org/10.1016/j.arcontrol.2021.10.014

Arduino. (2024). Servo Reference. Retrieved September 24, 2024, from https://www.arduino.cc/reference/en/libraries/servo/writemicroseconds/

Bhagat, S., Banerjee, H., Tse, Z. T. H., & Ren, H. (2019, January). Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges. https://doi.org/10.3390/robotics8010004
Meta-studie.

Calisti, M., Picardi, G., & Laschi, C. (2017, May). Fundamentals of soft robot locomotion. https://doi.org/10.1098/rsif.2017.0101

Gillespie, M. T., Best, C. M., Townsend, E. C., Wingate, D., & Killpack, M. D. (2018). Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, 39–45. https://doi.org/10.1109/ROBOSOFT.2018.8404894

Hawkes, E. W., Blumenschein, L. H., Greer, J. D., & Okamura, A. M. (2017). A soft robot that navigates its environment through growth. *Science Robotics*, *2*(8), 1–8. https://doi.org/10.1126/scirobotics.aan3028

Kim, D., Kim, S.-H., Kim, T., Kang, B. B., Lee, M., Park, W., Ku, S., Kim, D., Kwon, J., Lee, H., Bae, J., Park, Y.-L., Cho, K.-J., & Jo, S. (2021). Review of machine learning methods in soft robotics (G. Gu, Ed.). *PLOS ONE*, *16*(2), e0246102. https://doi.org/10.1371/journal.pone.0246102

Kresling, B. (2008). Natural twist buckling in shells: from the hawkmoth's bellows to the deployable Kresling-pattern and cylindrical Miura-ori. *Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: "Spanning Nano to Mega"*, (May), 1–4.

Landau, I. D., Lozano, R., M'Saad, M., & Karimi, A. (2011). *Adaptive Control*. Springer London. https://doi.org/10.1007/978-0-85729-664-1

Lao, D., Quan, Y., Wang, F., & Liu, Y. (2023). Error Modeling and Parameter Calibration Method for Industrial Robots Based on 6-DOF Position and Orientation. *Applied Sciences (Switzerland)*, *13*(19), 10901. https://doi.org/10.3390/app131910901

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., & Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, *58*, 52–68. https://doi.org/10.1016/j.inffus.2019.12.004

Li, R. J., & Han, Z. Z. (2005). Survey of iterative learning control. *Kongzhi yu Juece/Control and Decision*, *20*(9), 961–966. https://doi.org/10.1109/mcs.2006.1636313

Li, Z., & Hoiem, D. (2018). Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(12), 2935–2947. https://doi.org/10.1109/TPAMI.2017.2773081

Marshall, S. A. (1978). Steady-state and Transient Behaviour of Control Systems. In *Introduction to control theory* (pp. 73–104). Macmillan Education UK. https://doi.org/10.1007/978-1-349-15910-9_5

Masana, R., Dalaq, A. S., Khazaaleh, S., & Daqaq, M. F. (2024). The Kresling Origami Spring : A Review and Assessment. *Smart Materials and Structures*, *33*. https://doi.org/10.1088/1361-665X/AD2F6F

Masana, R., & Daqaq, M. F. (2019). Equilibria and bifurcations of a foldable paper-based spring inspired by Kresling-pattern origami. *Physical Review E*, *100*(6), 063001. https://doi.org/10.1103/PhysRevE.100.063001

Pagano, A., Yan, T., Chien, B., Wissa, A., & Tawfick, S. (2017). A crawling robot driven by multi-stable origami. *Smart Materials and Structures*, *26*(9). https://doi.org/10.1088/1361-665X/aa721e

Rob Tillaart. (2024). Arduino library for AS5600 magnetic rotation meter. Retrieved September 24, 2024, from https://github.com/RobTillaart/AS5600

Tang, J., & Wei, F. (2022). Miniaturized Origami Robots: Actuation Approaches and Potential Applications. *Macromolecular Materials and Engineering*, *307*(2), 2100671. https://doi.org/10.1002/mame.202100671

Thuruthel, T. G., Falotico, E., Renda, F., & Laschi, C. (2019). Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators. *IEEE Transactions on Robotics*, *35*(1), 124–134. https://doi.org/10.1109/TRO.2018.2878318

Wang, Z., Chen, C., & Dong, D. (2020). Lifelong Incremental Reinforcement Learning with Online Bayesian Inference. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(8), 4003–4016. https://doi.org/10.1109/TNNLS.2021.3055499

Ze, Q., Wu, S., Nishikawa, J., Dai, J., Sun, Y., Leanza, S., Zemelka, C., Novelino, L. S., Paulino, G. H., & Zhao, R. R. (2022). Soft robotic origami crawler. *Science Advances*, *8*(13), 7834. https://doi.org/10.1126/sciadv.abm7834

Appendix A
Software implementation

The supervised learning implementation for target adaptations is divided into two key components: the nominal control of the robot, programmed in C++ and executed by the Arduino board, and the target adaptation, implemented in Python and managed by an external computer. The adapter is implemented and trained using Keras with the PyTorch backend.

A. Nominal Control on the Arduino

Upon startup, the Arduino initializes a connection with the external computer and directs the servo to move to its minimum position. The Arduino allows some time for the servo to reach this home position and then reads a first measurement from the angle sensor, setting this as the reference $\theta = 0$. During the control loop, the servo reads θ (relative to the reference) and $\dot{\theta}$ received from the sensor, and reads the target θ_d received from the external computer. Based on these measurements, the PID controller computes the value to be written to the servo, but before writing it to the servo, a ceiling is imposed on it to protect the servo from exceeding its mechanical limit (Section V-A). The Arduino allows some time for the servo to act and then reads the measurements from the angle sensor again. Finally, it sends the tuple containing the resulting state and target to the external computer.

B. Target Adaptation on the External Computer

The target adaptation and supervised learning are implemented exclusively on the external computer through the established connection with the Arduino. The program is divided into several modules or "threads" that run simultaneously. This

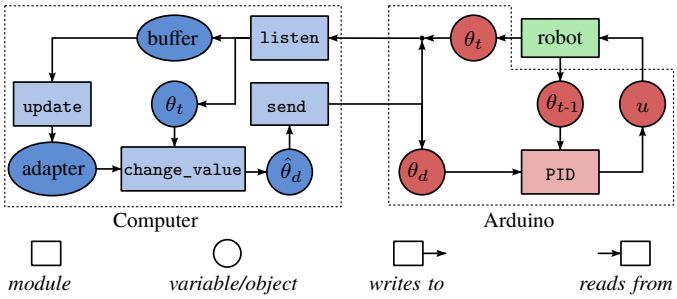


Fig. A.1: The interaction between the modules (rectangles) and shared variables (ovals) of the computer (blue) and Arduino (red).

way, values that need to be sent or read can be updated in the background upon request, without delaying the operations of other modules.

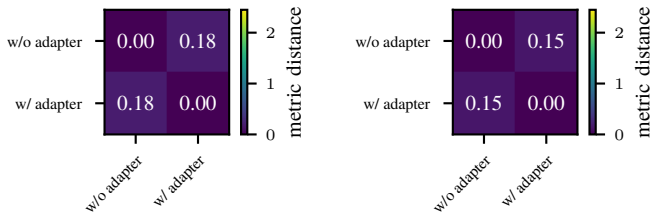
Figure A.1 provides an overview of the communication between computer and Arduino, and their inner interactions between various modules. For example, the sole task of the `send` thread is to read the value of a global variable representing the target state and send this to the Arduino. The `listen` thread reads the tuple sent in return by the Arduino and assigns its contents to the respective shared variables for the resulting state and target. It then adds this tuple to the buffer used for updating the adapter model. The `update` thread is responsible for (re)training the adapter. It creates a copy of the adapter, which can then be trained in the background while the original adapter remains functional. Once training is completed, the copied adapter's weights are assigned to the original adapter. The `change_value` thread is responsible for setting the *true* target in the control loop. Every iteration, it constructs an input feature $[\theta, \dot{\theta}, \theta_d, 0]$ for the adapter. The resulting Δ is added to the target, which is read by the `send` thread and forwarded to the Arduino. The buffer is maintained by storing the tuples received from the Arduino. All threads aim to operate at 50Hz.

Appendix B
KOS Reference Recordings

Before and during each experiment, with intervals of 30min, a minute of reference data was recorded. These measurements serve as a basis for comparing the relevant KOSs involved in each experiment, since they are supposed to be reasonably similar to each other (Section VIII-C). The measurements presented are the numeric values of these evaluations, as well as visual representations of the distance in the normalized metric space between the vectors represented by the evaluation of the first reference minute, i.e. *before* the experiment has started.

Values used for normalizing the components of the metric vector were: 5 (theoretical maximum rise time), 5 (theoretical maximum settling time), 25 (physical maximum overshoot for the actuation arm), 25 (physical maximum error for the actuation arm), 480 (maximum servo velocity), 1800 (maximum control output).

A. Reference Minutes Slow-Agressive PID Controller



(a) *slow-aggressive* PID (b) *under-tuned* PID

Fig. B.1: Distance in the metric space between the KOSs at the start of the experiment.

TABLE B.I: Numerical values of the reference minute metrics for the *slow-aggressive* PID experiment.

(a) Baseline controller

Metric	0	30	60	90
MRT [s]	0.12	0.15	0.12	0.14
MST [s]	3.41	3.26	2.02	1.61
MO [deg]	7.33	5.54	5.60	5.07
MAE [deg]	4.14	2.23	1.96	0.98
MAV [deg/s]	49.53	27.23	24.95	12.44
NTV [µs]	26.61	17.70	15.48	7.91

(b) (Disabled) target adapter

Metric	0	30	60	90
MRT [s]	0.15	0.15	0.14	0.16
MST [s]	2.52	3.49	3.44	3.34
MO [deg]	7.95	6.55	5.98	5.04
MAE [deg]	3.73	2.99	2.83	2.01
MAV [deg/s]	40.46	39.57	37.83	29.24
NTV [µs]	23.42	24.42	23.73	17.13

B. Reference Minutes Under-Tuned PID Controller

TABLE B.II: Numerical values of the reference minute metrics for the *under-tuned* PID experiment.

(a) Baseline controller

Metric	0	30	60	90	120
MRT [s]	2.05	2.22	2.23	2.24	2.19
MST [s]	2.61	3.26	3.31	3.38	3.49
MO [deg]	0.05	0.19	0.14	0.03	0.25
MAE [deg]	1.29	1.36	1.32	1.32	1.27
MAV [deg/s]	3.77	4.03	3.61	3.82	3.94
NTV [µs]	1.65	1.68	1.63	1.67	1.70

(b) (Disabled) target adapter

Metric	0	30	60	90	120
MRT [s]	1.30	1.45	1.48	1.61	1.69
MST [s]	2.50	2.60	2.60	2.96	3.08
MO [deg]	0.25	0.02	0.10	0.12	0.12
MAE [deg]	0.87	0.84	0.85	0.93	0.96
MAV [deg/s]	4.44	3.88	3.67	3.90	3.95
NTV [µs]	1.54	1.44	1.42	1.47	1.54

C. Reference Minutes 0.25–0.25 & 0.5–0.5 Perforations

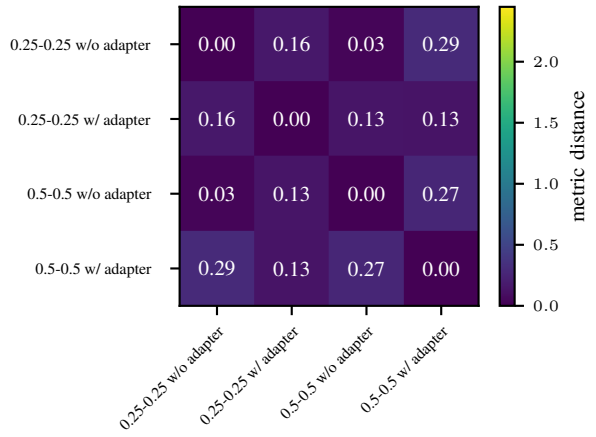


Fig. B.2: Distance in the metric space between the KOSs at the start of the 0.25–0.25mm and 0.5–0.5mm perforation experiment.

TABLE B.III: Numerical values of the reference minute metrics for the 0.25–0.25mm and 0.5–0.5mm perforation experiment.

(a) Baseline controller

Metric	0
MRT [s]	0.73
MST [s]	2.04
MO [deg]	0.80
MAE [deg]	0.80
MAV [deg/s]	4.97
NTV [µs]	4.85

(b) (Disabled) target adapter

Metric	0
MRT [s]	1.68
MST [s]	2.96
MO [deg]	0.03
MAE [deg]	1.05
MAV [deg/s]	4.11
NTV [µs]	4.45

D. Reference Minutes 1.5–0.5 & 2–0.5 Perforations

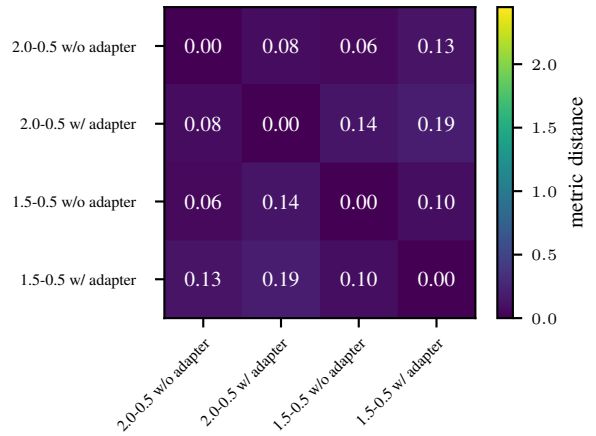


Fig. B.3: Distance in the metric space between the KOSs at the start of the 1.5–0.5mm and 2–0.5mm perforation experiment.

TABLE B.IV: Numerical values of the reference minute metrics for the 1.5–0.5mm and 2–0.5mm perforation experiment.

(a) Baseline controller

Metric	0	30	60	90	120
MRT [s]	0.24	0.28	0.32	0.55	0.64
MST [s]	0.90	1.00	1.51	1.87	2.17
MO [deg]	2.82	1.50	1.33	0.90	1.09
MAE [deg]	0.44	0.48	0.54	0.69	0.70
MAV [deg/s]	7.75	6.09	5.65	8.30	9.47
NTV [µs]	6.04	5.06	5.24	7.00	7.77

(b) (Disabled) target adapter

Metric	0	30	60	90	120
MRT [s]	0.53	0.55	0.47	0.33	0.52
MST [s]	1.10	1.32	1.62	1.25	1.59
MO [deg]	1.21	1.61	1.16	1.43	1.35
MAE [deg]	0.48	0.56	0.59	0.50	0.52
MAV [deg/s]	6.62	6.45	6.08	6.62	6.49
NTV [µs]	5.15	5.75	5.10	5.62	5.53

E. Reference Minutes Crawling Gait

TABLE B.V: Numerical values of the reference minute metrics for the crawling gait experiment.

	(a) Baseline controller		(b) (Disabled) pretrained target adapter		(c) (Disabled) online target adapter	
Metric	**0**	**30**	**0**	**30**	**0**	**30**
MRT [s]	0.27	1.37	0.40	0.30	0.29	0.31
MST [s]	1.27	3.35	1.61	0.52	1.18	1.28
MO [deg]	1.42	-0.10	2.99	2.13	1.43	2.58
MAE [deg]	0.61	1.11	0.70	0.46	0.59	0.56
MAV [deg/s]	6.80	5.00	6.06	6.72	6.71	8.19
NTV [µs]	5.83	5.29	5.69	5.48	5.98	7.17

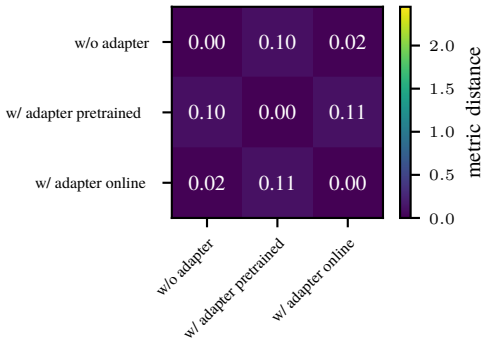


Fig. B.4: Distance in the metric space between the KOSs at the start of the crawling gait experiment.

Appendix C
Extended Test Results

A. *Slow-Aggressive PID Controller*

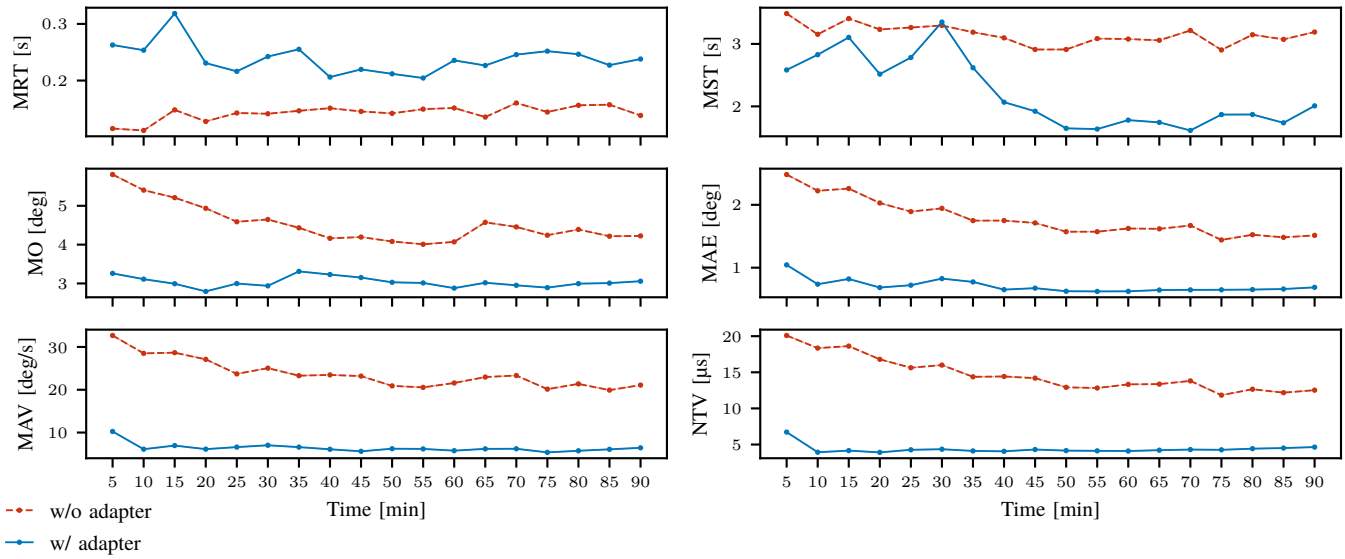


Fig. C.1: Metrics evolution for the *slow-aggressive* controller.

TABLE C.I: Numerical values of the metrics for the *slow-aggressive* controller.

(a) Baseline controller

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.12	0.11	0.15	0.13	0.14	0.14	0.15	0.15	0.15	0.14	0.15	0.15
MST [s]	3.49	3.15	3.41	3.23	3.26	3.29	3.19	3.10	2.91	2.91	3.09	3.08
MO [deg]	5.81	5.40	5.21	4.94	4.59	4.65	4.43	4.16	4.19	4.08	4.01	4.07
MAE [deg]	2.48	2.22	2.26	2.03	1.89	1.94	1.75	1.75	1.71	1.57	1.57	1.62
MAV [deg/s]	32.68	28.51	28.66	27.10	23.71	25.05	23.29	23.47	23.19	20.90	20.55	21.58
NTV [µs]	20.09	18.35	18.62	16.80	15.63	16.00	14.37	14.42	14.19	12.92	12.82	13.33

Metric	65	70	75	80	85	90
MRT [s]	0.14	0.16	0.14	0.16	0.16	0.14
MST [s]	3.06	3.22	2.90	3.15	3.07	3.19
MO [deg]	4.57	4.46	4.24	4.39	4.22	4.22
MAE [deg]	1.62	1.67	1.44	1.52	1.48	1.51
MAV [deg/s]	22.97	23.33	20.14	21.37	19.91	21.09
NTV [µs]	13.36	13.81	11.83	12.66	12.18	12.53

(b) Target adapter

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.26	0.25	0.32	0.23	0.22	0.24	0.26	0.21	0.22	0.21	0.20	0.24
MST [s]	2.58	2.83	3.10	2.52	2.78	3.35	2.62	2.07	1.92	1.65	1.64	1.78
MO [deg]	3.26	3.11	2.99	2.79	3.00	2.94	3.31	3.23	3.15	3.03	3.01	2.88
MAE [deg]	1.04	0.74	0.82	0.68	0.72	0.83	0.77	0.65	0.67	0.63	0.62	0.62
MAV [deg/s]	10.25	6.11	6.96	6.12	6.60	7.03	6.59	6.09	5.62	6.22	6.18	5.77
NTV [µs]	6.72	3.93	4.16	3.91	4.27	4.35	4.11	4.06	4.31	4.16	4.12	4.10

Metric	65	70	75	80	85	90
MRT [s]	0.23	0.25	0.25	0.25	0.23	0.24
MST [s]	1.74	1.61	1.87	1.87	1.74	2.01
MO [deg]	3.02	2.95	2.89	3.00	3.01	3.06
MAE [deg]	0.64	0.65	0.65	0.65	0.66	0.69
MAV [deg/s]	6.19	6.21	5.36	5.74	6.08	6.43
NTV [µs]	4.22	4.30	4.27	4.43	4.51	4.65

B. *Under-Tuned PID Controller*

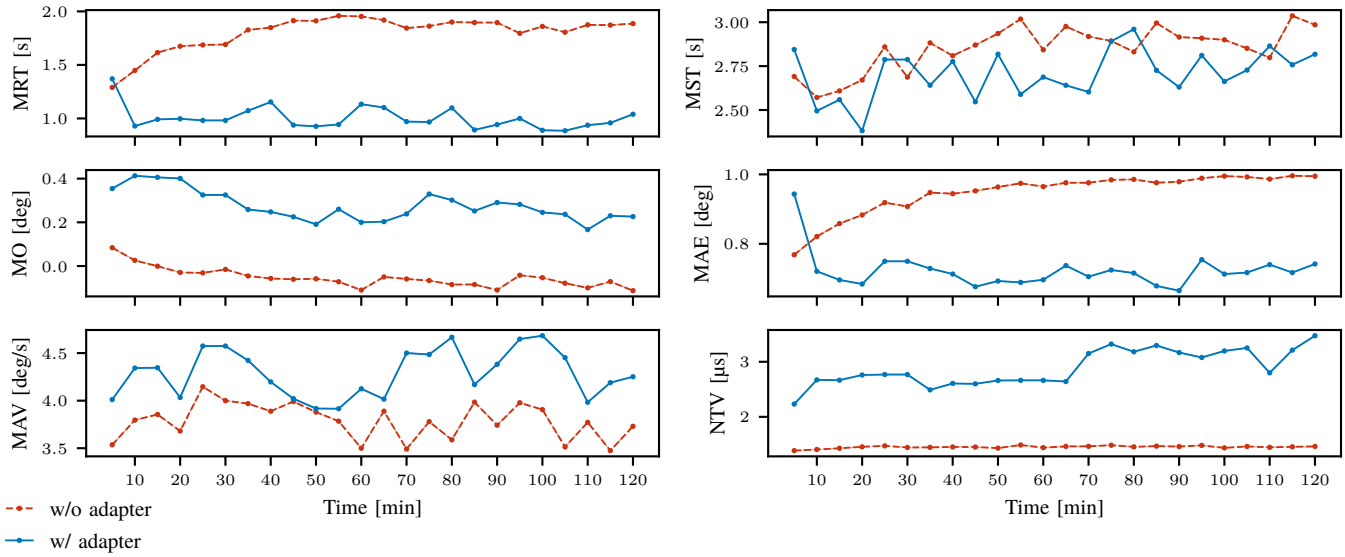


Fig. C.2: Metrics evolution for the *under-tuned* controller.

TABLE C.II: Numerical values of the metrics for the *under-tuned* controller.

(a) Baseline controller

Metric	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**
MRT [s]	1.29	1.45	1.61	1.67	1.69	1.69	1.83	1.85	1.91	1.91	1.96	1.95
MST [s]	2.69	2.57	2.61	2.67	2.86	2.69	2.88	2.81	2.87	2.94	3.02	2.84
MO [deg]	0.08	0.03	-0.00	-0.03	-0.03	-0.02	-0.05	-0.06	-0.06	-0.06	-0.07	-0.11
MAE [deg]	0.77	0.82	0.86	0.88	0.92	0.91	0.95	0.94	0.95	0.96	0.97	0.96
MAV [deg/s]	3.54	3.80	3.86	3.68	4.15	4.00	3.97	3.89	3.99	3.88	3.78	3.50
NTV [µs]	1.39	1.41	1.43	1.46	1.47	1.44	1.45	1.45	1.45	1.43	1.49	1.44
Metric	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**	**105**	**110**	**115**	**120**
MRT [s]	1.92	1.84	1.86	1.90	1.90	1.90	1.80	1.86	1.81	1.87	1.87	1.89
MST [s]	2.98	2.92	2.89	2.83	3.00	2.92	2.91	2.90	2.85	2.80	3.04	2.99
MO [deg]	-0.05	-0.06	-0.07	-0.08	-0.08	-0.11	-0.04	-0.05	-0.08	-0.10	-0.07	-0.11
MAE [deg]	0.98	0.98	0.98	0.99	0.98	0.98	0.99	0.99	0.99	0.99	1.00	0.99
MAV [deg/s]	3.89	3.49	3.78	3.59	3.98	3.74	3.98	3.91	3.51	3.77	3.47	3.73
NTV [µs]	1.46	1.46	1.49	1.46	1.47	1.46	1.48	1.44	1.46	1.45	1.46	1.46

(b) Target adapter

Metric	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**
MRT [s]	1.37	0.93	0.99	1.00	0.98	0.98	1.07	1.15	0.94	0.93	0.94	1.13
MST [s]	2.84	2.49	2.56	2.38	2.79	2.79	2.64	2.78	2.55	2.82	2.59	2.69
MO [deg]	0.35	0.41	0.41	0.40	0.33	0.33	0.26	0.25	0.23	0.19	0.26	0.20
MAE [deg]	0.94	0.72	0.70	0.68	0.75	0.75	0.73	0.71	0.68	0.69	0.69	0.70
MAV [deg/s]	4.01	4.34	4.35	4.03	4.57	4.57	4.42	4.20	4.02	3.92	3.92	4.12
NTV [µs]	2.23	2.67	2.67	2.76	2.77	2.77	2.49	2.61	2.60	2.66	2.66	2.66
Metric	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**	**105**	**110**	**115**	**120**
MRT [s]	1.10	0.97	0.97	1.10	0.89	0.94	1.00	0.89	0.89	0.94	0.96	1.04
MST [s]	2.64	2.60	2.89	2.96	2.73	2.63	2.81	2.66	2.73	2.86	2.76	2.82
MO [deg]	0.20	0.24	0.33	0.30	0.25	0.29	0.28	0.25	0.24	0.17	0.23	0.23
MAE [deg]	0.74	0.71	0.72	0.72	0.68	0.67	0.75	0.71	0.72	0.74	0.72	0.74
MAV [deg/s]	4.02	4.50	4.49	4.67	4.17	4.38	4.65	4.68	4.45	3.98	4.19	4.25
NTV [µs]	2.64	3.15	3.32	3.18	3.30	3.17	3.08	3.20	3.25	2.80	3.21	3.47

C. 0.25–0.25 & 0.5–0.5 Perforations

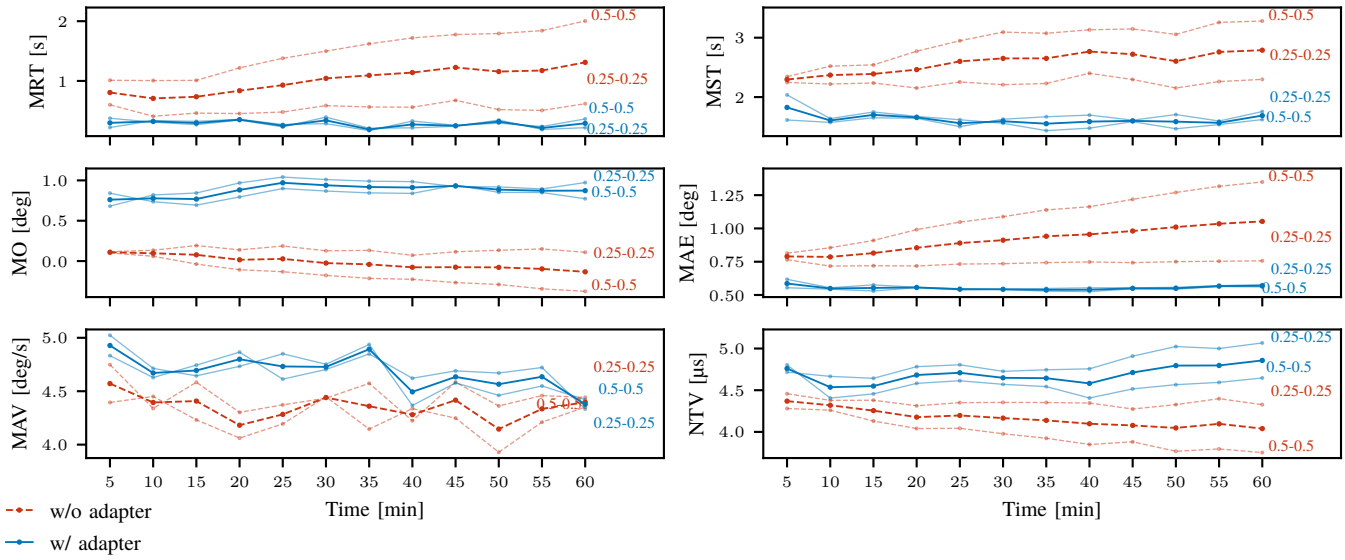


Fig. C.3: Metrics evolution for the 25–25 and 50–50mm perforations.

TABLE C.III: Numerical values of the metrics for the 25–25 and 50–50mm perforations.

(a) 0.25–0.25mm Baseline controller

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.60	0.41	0.46	0.45	0.48	0.59	0.56	0.56	0.68	0.52	0.51	0.62
MST [s]	2.25	2.22	2.24	2.15	2.25	2.21	2.23	2.40	2.30	2.15	2.26	2.30
MO [deg]	0.12	0.13	0.19	0.14	0.19	0.13	0.13	0.07	0.12	0.13	0.15	0.11
MAE [deg]	0.77	0.72	0.72	0.72	0.73	0.74	0.74	0.75	0.74	0.75	0.75	0.76
MAV [deg/s]	4.75	4.34	4.58	4.30	4.37	4.43	4.57	4.22	4.58	4.36	4.46	4.44
NTV [µs]	4.46	4.38	4.38	4.31	4.35	4.35	4.35	4.35	4.27	4.33	4.40	4.33

(b) 0.25–0.25mm Target adapter

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.38	0.31	0.27	0.35	0.23	0.39	0.21	0.22	0.24	0.35	0.19	0.22
MST [s]	1.61	1.57	1.65	1.64	1.50	1.63	1.67	1.70	1.61	1.71	1.60	1.76
MO [deg]	0.68	0.82	0.84	0.97	1.04	1.01	0.99	0.98	0.92	0.92	0.89	0.97
MAE [deg]	0.55	0.54	0.53	0.56	0.54	0.54	0.55	0.55	0.55	0.56	0.57	0.58
MAV [deg/s]	4.83	4.63	4.74	4.87	4.61	4.70	4.85	4.62	4.69	4.67	4.72	4.33
NTV [µs]	4.72	4.67	4.64	4.78	4.80	4.73	4.74	4.76	4.91	5.02	5.00	5.07

(c) 0.5–0.5mm Baseline controller

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	1.01	1.01	1.01	1.22	1.38	1.50	1.62	1.72	1.78	1.79	1.84	2.00
MST [s]	2.34	2.52	2.54	2.77	2.95	3.09	3.07	3.13	3.15	3.06	3.26	3.28
MO [deg]	0.10	0.06	-0.04	-0.11	-0.13	-0.18	-0.21	-0.23	-0.27	-0.29	-0.34	-0.38
MAE [deg]	0.81	0.85	0.91	0.99	1.05	1.09	1.14	1.16	1.22	1.27	1.32	1.35
MAV [deg/s]	4.39	4.45	4.23	4.06	4.19	4.45	4.15	4.34	4.25	3.93	4.21	4.35
NTV [µs]	4.28	4.26	4.13	4.04	4.04	3.98	3.92	3.85	3.88	3.77	3.80	3.75

(d) 0.5–0.5mm Target adapter

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.22	0.34	0.32	0.36	0.27	0.29	0.17	0.33	0.26	0.29	0.24	0.36
MST [s]	2.04	1.64	1.75	1.67	1.62	1.56	1.44	1.48	1.59	1.47	1.54	1.62
MO [deg]	0.84	0.74	0.69	0.79	0.90	0.87	0.85	0.84	0.94	0.85	0.85	0.77
MAE [deg]	0.62	0.55	0.58	0.56	0.55	0.54	0.53	0.53	0.55	0.54	0.56	0.56
MAV [deg/s]	5.02	4.71	4.64	4.73	4.85	4.75	4.94	4.37	4.58	4.46	4.55	4.42
NTV [µs]	4.80	4.41	4.46	4.58	4.61	4.57	4.54	4.41	4.52	4.57	4.59	4.65

D. 1.5–0.5 & 2–0.5 Perforations

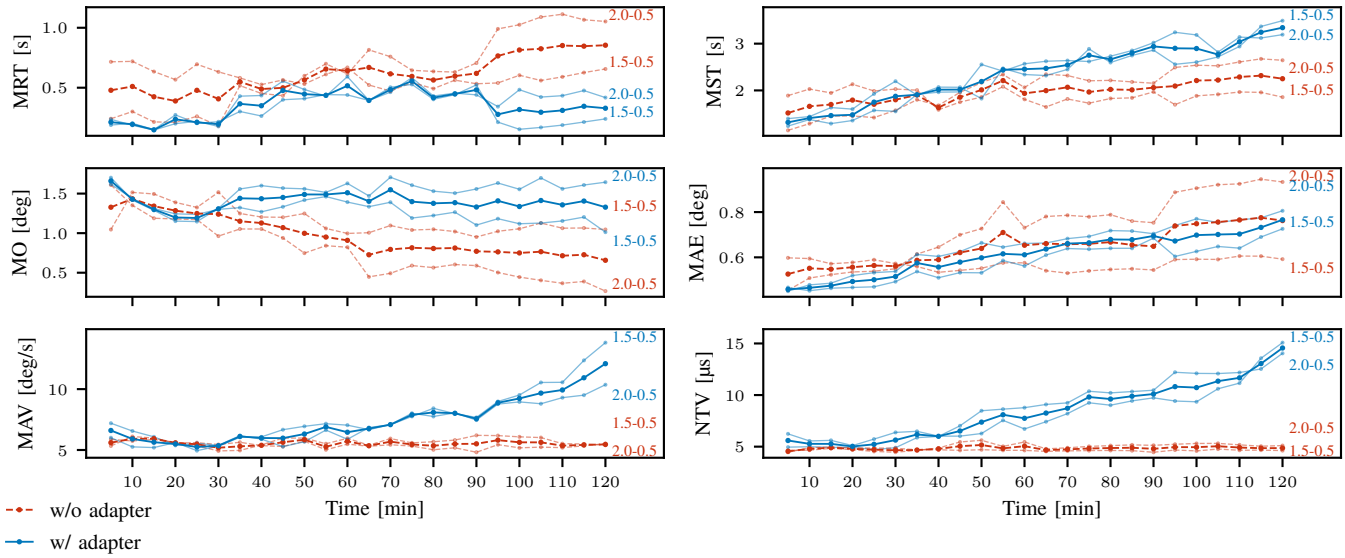


Fig. C.4: Metrics evolution for the 150–50 and 200–50mm perforations.

TABLE C.IV: Numerical values of the metrics for the 150–50 and 200–50mm perforations.

(a) 1.5–0.5mm Baseline controller

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.72	0.72	0.63	0.57	0.70	0.63	0.58	0.53	0.56	0.53	0.61	0.67
MST [s]	1.89	2.03	1.95	2.13	1.99	2.03	2.01	1.59	1.75	1.86	2.08	1.81
MO [deg]	1.05	1.52	1.50	1.39	1.33	1.52	1.25	1.20	1.20	1.25	1.06	1.00
MAE [deg]	0.60	0.59	0.57	0.58	0.59	0.57	0.56	0.53	0.54	0.55	0.58	0.58
MAV [deg/s]	5.33	6.11	5.98	5.57	5.60	5.45	5.63	5.36	5.29	5.72	5.02	5.48
NTV [µs]	4.46	4.90	4.96	4.87	4.87	4.84	4.70	4.70	4.64	4.70	4.65	4.64

Metric	65	70	75	80	85	90	95	100	105	110	115	120
MRT [s]	0.52	0.47	0.54	0.49	0.56	0.53	0.54	0.60	0.56	0.59	0.63	0.66
MST [s]	1.65	1.82	1.73	1.83	1.84	1.97	1.70	1.88	1.92	1.97	1.96	1.86
MO [deg]	1.01	1.10	1.04	1.05	1.02	0.95	1.02	1.06	1.13	1.06	1.07	1.05
MAE [deg]	0.54	0.53	0.54	0.55	0.55	0.54	0.59	0.59	0.59	0.61	0.61	0.59
MAV [deg/s]	5.31	5.43	5.32	5.02	5.18	4.81	5.44	5.18	5.23	5.19	5.34	5.52
NTV [µs]	4.54	4.59	4.62	4.63	4.61	4.46	4.67	4.59	4.76	4.65	4.66	4.62

(b) 1.5–0.5mm Target adapter

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.19	0.20	0.15	0.27	0.20	0.22	0.30	0.26	0.40	0.41	0.44	0.44
MST [s]	1.40	1.44	1.63	1.60	1.93	2.20	1.90	2.07	2.07	1.83	2.47	2.34
MO [deg]	1.62	1.44	1.31	1.24	1.24	1.30	1.32	1.27	1.33	1.42	1.46	1.39
MAE [deg]	0.47	0.45	0.46	0.47	0.47	0.49	0.54	0.51	0.53	0.53	0.59	0.56
MAV [deg/s]	7.20	6.56	6.08	5.38	5.55	5.39	6.18	5.90	5.28	5.68	6.64	5.88
NTV [µs]	6.24	5.55	5.61	5.10	5.73	6.38	6.49	6.02	6.02	6.27	7.55	6.72

Metric	65	70	75	80	85	90	95	100	105	110	115	120
MRT [s]	0.39	0.50	0.53	0.40	0.44	0.52	0.21	0.15	0.17	0.19	0.21	0.24
MST [s]	2.32	2.45	2.89	2.60	2.75	2.86	2.56	2.61	2.72	2.94	3.37	3.49
MO [deg]	1.33	1.39	1.19	1.22	1.27	1.10	1.18	1.12	1.13	1.15	1.20	1.01
MAE [deg]	0.61	0.64	0.64	0.64	0.64	0.68	0.60	0.63	0.65	0.64	0.69	0.73
MAV [deg/s]	6.85	7.07	7.76	8.42	7.98	7.67	8.98	9.52	10.55	10.56	12.36	13.82
NTV [µs]	7.42	8.20	9.25	9.02	9.43	9.73	9.42	9.35	10.61	11.16	13.57	15.08

TABLE C.IV: Numerical values of the metrics for the 150–50 and 200–50mm perforations, *continued*.

(c) 2–0.5mm Baseline controller

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.24	0.30	0.21	0.21	0.26	0.18	0.52	0.45	0.43	0.60	0.70	0.61
MST [s]	1.15	1.29	1.45	1.45	1.42	1.57	1.80	1.68	1.96	2.17	2.34	2.06
MO [deg]	1.61	1.35	1.19	1.18	1.17	0.96	1.05	1.05	0.94	0.75	0.84	0.82
MAE [deg]	0.45	0.51	0.52	0.54	0.54	0.55	0.61	0.65	0.70	0.73	0.84	0.73
MAV [deg/s]	5.87	5.74	5.87	5.60	5.34	4.93	4.97	5.41	5.98	5.98	5.51	5.96
NTV [µs]	4.64	4.66	4.84	4.71	4.54	4.46	4.64	4.87	5.45	5.62	5.02	5.45

Metric	65	70	75	80	85	90	95	100	105	110	115	120
MRT [s]	0.82	0.76	0.65	0.64	0.63	0.71	0.99	1.03	1.09	1.11	1.07	1.05
MST [s]	2.35	2.32	2.21	2.22	2.18	2.15	2.49	2.54	2.53	2.61	2.67	2.64
MO [deg]	0.45	0.49	0.59	0.56	0.60	0.59	0.50	0.44	0.40	0.37	0.39	0.27
MAE [deg]	0.78	0.79	0.78	0.79	0.76	0.75	0.89	0.91	0.92	0.92	0.95	0.93
MAV [deg/s]	5.39	5.94	5.59	5.68	5.81	6.20	6.18	6.08	6.02	5.52	5.49	5.39
NTV [µs]	4.79	4.89	5.02	5.10	5.17	5.13	5.23	5.31	5.32	5.17	5.04	5.10

(d) 2–0.5mm Target adapter

Metric	5	10	15	20	25	30	35	40	45	50	55	60
MRT [s]	0.24	0.19	0.15	0.20	0.22	0.18	0.43	0.44	0.55	0.48	0.43	0.59
MST [s]	1.24	1.38	1.29	1.36	1.57	1.55	1.92	1.96	1.97	2.55	2.41	2.57
MO [deg]	1.70	1.42	1.29	1.15	1.15	1.31	1.56	1.60	1.57	1.56	1.51	1.63
MAE [deg]	0.45	0.48	0.49	0.52	0.53	0.54	0.61	0.60	0.63	0.66	0.65	0.66
MAV [deg/s]	6.01	5.26	5.20	5.62	4.95	5.31	6.05	6.08	6.67	6.94	7.16	7.04
NTV [µs]	4.95	4.99	4.93	4.94	4.72	4.90	5.87	5.99	7.02	8.48	8.63	8.78

Metric	65	70	75	80	85	90	95	100	105	110	115	120
MRT [s]	0.39	0.46	0.58	0.43	0.45	0.44	0.34	0.48	0.42	0.43	0.48	0.42
MST [s]	2.62	2.64	2.62	2.73	2.85	3.02	3.24	3.19	2.82	3.14	3.12	3.19
MO [deg]	1.47	1.71	1.61	1.53	1.51	1.56	1.63	1.55	1.70	1.56	1.61	1.64
MAE [deg]	0.66	0.68	0.69	0.72	0.72	0.70	0.74	0.77	0.75	0.77	0.77	0.81
MAV [deg/s]	6.67	7.09	8.02	7.75	8.05	7.46	8.77	8.95	8.80	9.30	9.50	10.36
NTV [µs]	9.09	9.24	10.37	10.21	10.34	10.47	12.21	12.10	12.08	12.18	12.53	14.04

E. Crawling Gait

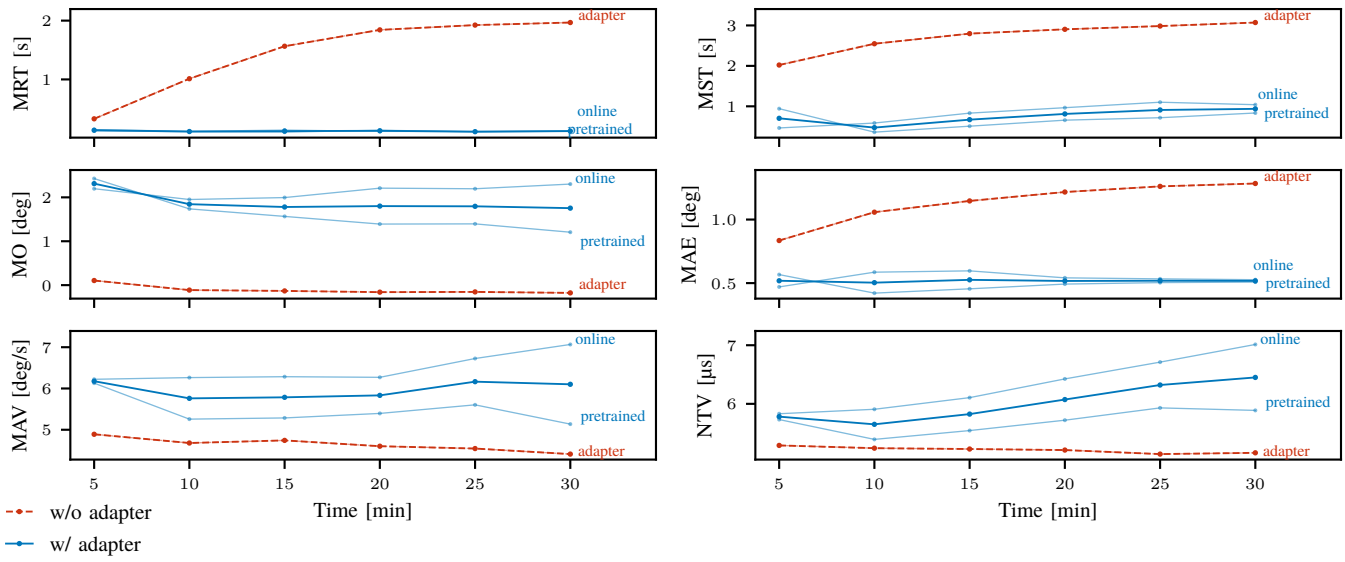


Fig. C.5: Metrics evolution for the crawling gait experiment.

TABLE C.V: Numerical values of the metrics for the crawling gait experiment.

(a) Baseline controller

Metric	5	10	15	20	25	30
MRT [s]	0.33	1.01	1.56	1.84	1.92	1.97
MST [s]	2.02	2.55	2.80	2.90	2.98	3.07
MO [deg]	0.10	-0.11	-0.13	-0.16	-0.15	-0.18
MAE [deg]	0.83	1.06	1.15	1.22	1.26	1.28
MAV [deg/s]	4.89	4.68	4.74	4.60	4.54	4.41
NTV [µs]	5.29	5.24	5.23	5.21	5.14	5.16

(b) Pretrained target adapter

Metric	5	10	15	20	25	30
MRT [s]	0.13	0.11	0.10	0.14	0.11	0.12
MST [s]	0.94	0.36	0.51	0.66	0.72	0.83
MO [deg]	2.43	1.74	1.57	1.39	1.40	1.21
MAE [deg]	0.57	0.42	0.46	0.49	0.50	0.51
MAV [deg/s]	6.13	5.26	5.29	5.40	5.60	5.14
NTV [µs]	5.73	5.39	5.54	5.72	5.93	5.89

(c) Online trained target adapter

Metric	5	10	15	20	25	30
MRT [s]	0.15	0.12	0.14	0.12	0.11	0.12
MST [s]	0.46	0.59	0.83	0.97	1.10	1.04
MO [deg]	2.20	1.95	2.00	2.21	2.20	2.30
MAE [deg]	0.47	0.59	0.60	0.54	0.53	0.53
MAV [deg/s]	6.22	6.26	6.29	6.27	6.73	7.07
NTV [µs]	5.83	5.91	6.11	6.43	6.71	7.01

Appendix D
KOS Design and Template

The KOSs used during the experiments are of type B1 and used the following design parameters, as related to the design map from Section II-C:

$$\begin{aligned} a_0 &\quad 25 \text{ mm} \\ a_0/b_0 &\quad 1.5 \\ \alpha_0 &\quad 28 \text{ degrees} \end{aligned}$$

The springs were constructed from $160\text{g}/\text{m}^2$ paper. A template for constructing this KOS can be found at the end of the document. The template uses the 1.5–0.5mm perforation.

Instructions:

1) Tear out the template along its outer contours.
2) Fold the Kresling pattern: The parallelogram's diagonals are valley folds, the sides are mountain folds.
3) Form a cylinder and glue the flaps at the ends to the insides of the cylinder on the opposite ends of the template.
4) Insert the extensions at the top and bottom of the cylinder through the slits in the two base hexagons.
5) Fold the extensions outward and glue them to the bases.
6) For extra strength, an extra (cardboard) hexagonal base can be glued to the top and bottom of the KOS.

Template

Valley fold Mountain fold Cut 1.5 - 0.5 perforation 0.25 - 1.75 perforation