Exploiting short-term memory in soft body dynamics as a computational resource

K. Nakajima1,2, T. Li3, H. Hauser3 and R. Pfeifer3

1The Hakubi Center for Advanced Research, and 2Department of Applied Analysis and Complex Dynamical Systems, Graduate School of Informatics, Kyoto University, 606-8501 Kyoto, Japan
3Artificial Intelligence Laboratory, Department of Informatics, University of Zurich, 8050 Zurich, Switzerland

Soft materials are not only highly deformable, but they also possess rich and diverse body dynamics. Soft body dynamics exhibit a variety of properties, including nonlinearity, elasticity and potentially infinitely many degrees of freedom. Here, we demonstrate that such soft body dynamics can be employed to conduct certain types of computation. Using body dynamics generated from a soft silicone arm, we show that they can be exploited to emulate functions that require memory and to embed robust closed-loop control into the arm. Our results suggest that soft body dynamics have a short-term memory and can serve as a computational resource. This finding paves the way towards exploiting passive body dynamics for control of a large class of underactuated systems.

1. Introduction

In recent years, soft materials have been increasingly used to incorporate flexible elements into robots’ bodies. The resulting machines, called soft robots, have significant advantages over traditional articulated robots owing to deformable morphology and safety in interaction [1]. They can adapt their morphology to unstructured environments, and carry and touch fragile objects without causing damage, which makes them applicable for rescue and human interactions, in particular care for the elderly, prosthetics and wearables [2,3]. In addition, they can generate diverse behaviours with simple types of actuation by partially outsourcing control to the morphological and material properties of their soft bodies [4], which is made possible by the tight coupling between control, body and environment [5,6]. In this paper, we build on these perspectives and add a novel advantage of soft bodies, demonstrating that they can be exploited as computational resources.

One of the major differences between rigid and soft bodies can be found in their body dynamics. Soft body dynamics usually exhibit a variety of properties, including nonlinearity, elasticity and potentially infinitely many degrees of freedom, which are difficult to reduce to lower dimensionality. In particular, their degrees of freedom are often larger than a number of actuators, which leads to a typical underactuated system [7], and this makes the soft body difficult to control with conventional frameworks. Here, we demonstrate that these properties can, in fact, be highly beneficial in that they can be employed for computation. Our approach is based on a machine learning technique called reservoir computing, which has a particular focus on real-time computing of time-varying input that provides an alternative to computational frameworks based on Turing machines [8–11]. By driving a high-dimensional dynamical system, typically referred to as the reservoir, with a low-dimensional input stream, transient dynamics are generated that operate as a type of temporal and finite kernel that facilitate the separation of input states [10,12]. If the dynamics involve enough nonlinearity and memory, emulating complex nonlinear dynamical systems only requires adding a linear, static readout from the high-dimensional state space of the reservoir. A number of different implementations for reservoirs have been proposed: for example, abstract
dynamical systems for echo state networks (ESNs) [8,9], or models of neurons for liquid state machines [10]. Implementations even include using the surface of water in a laminar state [13]. Lately, it has been demonstrated that nonlinear mass spring systems have the potential to serve as reservoirs as well [14,15], and this has been applied in a number of ways [16–18].

In this study, we establish a simple but powerful physical platform with a soft silicone arm and demonstrate, through a number of experiments, that the soft body dynamics can be used as a reservoir. In particular, we focus on the property of short-term memory [19–21], which is the ability to store information about recent input sequences in the transient dynamics of the reservoir. In neuroscience, this property has drawn attention as a mechanism to perform real-time computations on sensory input streams [22,23], which is a prerequisite for cognitive phenomena, such as planning and decision-making. We show that short-term memory also exists in the body dynamics of a soft silicone arm and, in particular, that it can be exploited to control the arm’s motions robustly in a closed-loop manner. In other words, the seemingly undesirable properties of soft body dynamics are no longer drawbacks for control but constitute core aspects of the system’s functionality.

2. Material and methods

2.1. A soft silicone arm as a computational resource

There have been several soft silicone arms proposed in the literature, which are inspired by the octopus [24–26]. In this paper, we use a soft silicone arm, which has a similar material characteristic to the one proposed in [24]. The platform consists of a soft silicone arm, its sensing and actuation systems, data processing via a PC, and a water tank containing fresh water as an underwater environment (figure 1). By rotating the base of the arm and generating body dynamics induced by the interaction between the underwater environment and the soft silicone material, we aim to show that the sensory timeseries that are reflected in the body dynamics can be exploited as part of a computational device. The unit of timestep used in this study is a sensing and actuation loop of the PC (this is approx. 0.03 s in physical time). Throughout this study, we observe the behaviour of the system from one side of the tank and use terminology, such as ‘left’ or ‘right’, with respect to this point of view.

The arm embeds 10 bend sensors within the silicone material (figure 1a and see also electronic supplementary material, figure S1). A bend sensor gives a base value when it is straight. If it bends in the ventral side, the sensor value is smaller, and if it bends in the dorsal side, the value is larger; the change in value reflects the degree of bend in each case. The sensors are embedded near the surface of the arm, with their ventral sides directed outwards. We numbered these sensors from the base towards the tip as s₁–s₁₀. The sensors are embedded alternately, with odd-numbered sensors on the right side of the arm and even-numbered sensors on the left side (figure 1a and electronic supplementary material, figure S1). The base of the arm can rotate left and right through the actuation of a servo motor. The motor commands sent from the PC are binary values, M = {0, 1}. If the command is 0 or 1, the motor is controlled to move from its current position towards the maximum right position (Lᵣight) or the maximum left position (Lᵣeft), respectively (figure 1b). The actual servo motor positions are also sent to the PC to monitor the current position of the base rotation θ(t). The positions Lᵣight and Lᵣeft were heuristically determined to avoid damaging the motor components. The values for |Lᵣight|

![Figure 1. Platform set-up for a soft silicone arm. (a) A soft silicone arm, which contains 10 bend sensors, is immersed underwater. Sensors are connected to a sensory board by the red wires. The wires are set as carefully as possible so as not to affect the arm motion. (b) Motor commands take binary states. When these commands are set to 0 (1), the base of the arm rotates to the right (left)-hand side towards Lᵣight (Lᵣeft). See the main text for details. (Online version in colour.)](https://rsif.royalsocietypublishing.org/content/11/101/20140437)
minimizing the error between the system output and the target output. The performance of the system output is evaluated by comparing with the target output for a new experimental trial (see the electronic supplementary material for details of the training procedures and the logistic regression).

We used three tasks to evaluate the computational power of our soft silicone arm with the focus on the property of short-term memory. Unlike a conventional computer, our system does not contain explicit memory storage; instead, the memory is expected to be implicitly included in the transient dynamics of the soft body. By assigning a task to the system that requires memory to be carried out and by evaluating its performance, we can characterize its memory capacity. Our first task is to construct a timer exploiting the soft body dynamics. Triggered by a cue sent at certain timesteps, the arm starts to move from \( L_{\text{right}} \) to \( L_{\text{left}} \). The system should output a pulse of predefined length by exploiting the body dynamics. To perform this task, the system has to be able to ‘recognize’ the duration of time that has passed since the cue was launched. This clearly requires memory. By increasing the desired pulse response, we systematically investigate the limits of the physical system to represent memory in its transient body dynamics.

The second task is to perform a closed-loop control exploiting the soft body dynamics. With a periodic square wave function, which switches its motor command from 0 to 1 and from 1 to 0 with a fixed period as a target function, we aim to evaluate the maximum length for the period of the square function that can be embedded in the system. In this task, the system should ‘recognize’ how much time has passed since the motor command switched from 0 to 1 (or from 1 to 0), and it should decide when to switch the motor command to the next position. Again, this task requires memory. Furthermore, this task also evaluates whether the soft body dynamics can be exploited as a computational resource to control the arm’s own motion. This is especially interesting as typically the complex dynamics of a soft body are the main obstacles to applying a classic control theoretic approach. Remarkably, in our proposed context, this property is beneficial because it can be exploited as a computational resource.

The third task is an emulation task of functions that require memory. A random binary input sequence is provided to the system, and by exploiting the generated soft body dynamics, the system should emulate two functions simultaneously: the first one is a function that reproduces past inputs with a given delay, and the second one is the N-bit parity checker. Emulations of these functions are commonly used as benchmark tasks to characterize the computational power of the system, and again, both functions require memory. In particular, these functions should be emulated using the same soft body dynamics at the same time, which points to another remarkable property of the approach (typically referred to as multitasking [14]).

In all three tasks, we are adjusting only the linear readout weights, which are fixed after learning, i.e. no memory is present in the readout. Hence, we can confirm that the required memory is purely owing to the property of the soft silicone arm. Unlike conventional computational units (e.g. artificial neural networks), our proposed set-up has a constraint owing to the specifications of the mechanical structure of the system, because inputs are transformed to the mechanical realm. For example, a drastic and frequent switching of the motor command can result in motor overheat and a total stop. We defined the presented tasks to evaluate the memory capacity of our system by taking these physical constraints into consideration. Accordingly, the input/output (I/O) setting in our system slightly differs in each task (see the electronic supplementary material for detailed information on the I/O setting for each task).

### 2.2. Dynamic property of the silicone arm

We here present the basic property of our arm motion and the step response. Figure 3a shows a typical arm motion when the motor command is switched from 0 to 1. The arm is initially set to \( L_{\text{right}} \), and at \( t = 0 \) it starts to move towards \( L_{\text{left}} \). The silicone arm shows characteristic body dynamics because of the interaction with the water (see electronic supplementary material, video S1). In particular, even when the base reaches the position of \( L_{\text{base}} \), the entire arm still shows transient dynamics. Figure 3 clearly shows that because the arm moves from right to left, the right side of the arm bends and the left side of the arm arches according to the water friction.

The dynamic behaviour of the arm can be captured by the responses of the sensors (figure 3b and electronic supplementary material, video S1). When the motor command switches from 0 to 1, \( \theta(t) \) takes about nine timesteps to reach \( \theta(t) = 1 \), which forms a physical constraint based on the motor and the mechanical structure of our platform (figure 3b, upper plot and electronic supplementary material, video S1). When the motor command is switched from 0 to 1, all the odd-numbered sensors start to show smaller values than those shown before the motion generation. They take the local minimum at a different timestep, then gradually approach their resting states (figure 3b, middle plot and electronic supplementary material, video S1). Because the arm is passive, the movement of the base rotation propagates from the base towards the tip at a certain velocity. For example, \( s1 \) seems to show a direct reflection of the motor actuation because it is embedded close to the base. This effect can be confirmed by checking the local minimum of the sensory response of \( s1 \) at around timestep 9, which is the same timestep at which the motor rotation stops. For even-numbered sensors, although all sensors show larger values than the values before the motion generation, some sensors (e.g. \( s6, s8 \) and \( s10 \)) show a smaller value in some timesteps owing to inertia caused by the immediate bend in the left side of the arm (figure 3b, lower plot and electronic supplementary material, video S1). This effect also seems to be propagating from the base towards the tip of the arm. All sensors reach a resting state at around 40 timesteps. In the resting state \( L_{\text{base}} \), the odd-numbered sensors show smaller values, and the even-numbered sensors show greater values than those shown before motion generation (figure 3b and...
3. Results

3.1. Timer task

Our first task is to emulate the function of a timer exploiting the body dynamics of the arm. The task has been chosen as it enables us to investigate systematically the memory inherently present in the soft body dynamics. One of the characteristic properties of our soft body is its transient dynamics during its motion from one state to another, e.g. moving from right to left. In this task, the arm is initially set to \( L_{\text{right}} \) and kept at this position. Triggered by the input at \( t_{\text{start}} \), the motor command switches from 0 to 1, when the rotation of the base generates the body dynamics (figure 3a). The timer task consists of producing an output pulse starting from \( \tau_{\text{ini}} \) timesteps after \( t_{\text{start}} \), which is \( \tau_{\text{timer}} \) timesteps in length, by exploiting the body dynamics during this transient single motion (see the electronic supplementary material, figure S2, for details). To perform this task, the system has to have a certain amount of memory. In other words, we can evaluate whether the sensory timeseries that reflects the transient dynamics during the motion from \( L_{\text{right}} \) to \( L_{\text{left}} \) contains sufficient information to recognize the duration of time since the trigger event by applying this task. A similar task was introduced in [8] to demonstrate the existence of short-term memory within an artificial recurrent neural network (e.g. ESN) [9,19]. To demonstrate that such a memory can be found and exploited in a real physical system, we applied this task employing the soft silicone arm. As explained earlier, our system output is generated by thresholding the weighted sum of the sensory values, and the weights are optimized with a simple logistic regression by using a dataset collected in the training phase (see the electronic supplementary material for details). We performed this experiment by varying \( \tau_{\text{ini}} \) and \( \tau_{\text{timer}} \) to investigate the relevance of these parameters to the system performance.

Figure 3c shows examples of the averaged system outputs for each \( \tau_{\text{timer}} \) (\( \tau_{\text{timer}} = 10, 15, 20, 30, 40 \) and 50) when \( \tau_{\text{ini}} \) is fixed to 9. Black lines show the target output and red lines show the averaged system outputs over 25 trials for each condition. Note that the averaged system outputs can take values in the range of \([0, 1]\). Figure 3d plots the average MSE over 25 trials with respect to each \( \tau_{\text{timer}} \) and \( \tau_{\text{ini}} \), varied from 1 to 50 and from 0 to 50, respectively.

Figure 3. Sensory response during the arm motion and performance for the timer task. (a) Snapshots showing a typical arm motion when the motor command is switched from 0 to 1 at \( t = 0 \) (i.e. movement from \( L_{\text{left}} \) to \( L_{\text{right}} \)). (b) Plots showing the dynamics of the motor command \( m(t) \) and the normalized base angle \( \theta(t) \) (the upper plot) and the corresponding sensory timeseries \( s(t) \) (the lower two plots). The middle and lower plots show the average sensory response curves for the odd- and even-numbered sensors, respectively. For each sensor, the sensory values are linearly scaled to make the sensory values to 1 when the arm is in \( L_{\text{right}} \) (the upper plot) and the corresponding sensory timeseries \( s(t) \). The timer task consists of producing an output pulse starting \( t_{\text{ini}} \) timesteps after \( t_{\text{start}} \), which is \( \tau_{\text{timer}} \) timesteps in length, by exploiting the body dynamics during this transient single motion (see the electronic supplementary material, figure S2, for details). To perform this task, the system has to have a certain amount of memory. In other words, we can evaluate whether the sensory timeseries that reflects the transient dynamics during the motion from \( L_{\text{right}} \) to \( L_{\text{left}} \) contains sufficient information to recognize the duration of time since the trigger event by applying this task. A similar task was introduced in [8] to demonstrate the existence of short-term memory within an artificial recurrent neural network (e.g. ESN) [9,19]. To demonstrate that such a memory can be found and exploited in a real physical system, we applied this task employing the soft silicone arm. As explained earlier, our system output is generated by thresholding the weighted sum of the sensory values, and the weights are optimized with a simple logistic regression by using a dataset collected in the training phase (see the electronic supplementary material for details). We performed this experiment by varying \( \tau_{\text{ini}} \) and \( \tau_{\text{timer}} \) to investigate the relevance of these parameters to the system performance.
time to propagate owing to the softness of the arm (figure 3a,b), and if \( \tau_{\text{ini}} \) is small, it is difficult to distinguish the sensory values from the values when the arm is stopped.

### 3.2. Closed-loop control task
We demonstrated in the previous task that we can use the sensory timeseries generated by the transient dynamics to construct a timer. By using the same property, in this second task, we aim to realize a closed-loop control of our soft silicone arm. That is, we aim to demonstrate that the arm’s body dynamics can be used to control its own motion. The target motor command sequence is a square wave in which the amplitude alternates at a steady frequency, between \( m(t) = 0 \) and \( 1 \), with the same duration of timesteps, \( \tau_{\text{square}} \) (see electronic supplementary material, figure S3a, for details). Similar to the process in the previous task, when the motor command is switched from 0 to 1 (or from 1 to 0), it should recognize the time length of \( \tau_{\text{square}} \) timesteps and switch the motor command from 1 to 0 (or from 0 to 1). Thus, it requires memory to fulfill this task. Recently, similar types of oscillatory motor command have been used to demonstrate the octopus-inspired swimming motion, called sculling, in a physical platform with an open-loop manner [27]. We aim to emulate this oscillatory wave pattern by using the sensory timeseries from the soft body and close the loop. This is realized by feeding back the system output generated by thresholding the weighted sum of the sensory timeseries as the next motor command to the system (see electronic supplementary material, figure S3b, for details). As with the previous task, we aim to emulate the target output only by adjusting the static linear readout weights.

Figure 4a shows an example of a timeseries with the motor commands and sensory values when the system is driven by the closed-loop control emulating a square function with \( \tau_{\text{square}} = 10 \). The timeseries of the motor command exactly overlaps with the target output, showing that the closed-loop control is successfully embedded (see also electronic supplementary material, video S2). For real-world applications, it is important to investigate whether the system is robust against external perturbation. We investigated the robustness of the system by applying a manual mechanical perturbation disturbing the arm motion (figure 4b,c and electronic supplementary material, video S3). We found that during the perturbation both the sensory timeseries and the system output were affected; however, after removing the disturbance, the system was able to recover immediately its original trajectory (electronic supplementary material, video S3). This can be confirmed by checking the timeseries of the motor commands and their corresponding sensory values, and it implies that our system is robust against external perturbations (figure 4b). Note that although

![Figure 4. Performance for the closed-loop control task.](image-url)

(a) Plot showing an example of the sensorimotor timeseries when the system is driven by the closed-loop control with a square function of \( \tau_{\text{square}} = 10 \). The system is initially driven by the target output until \( t = 50 \), and then the loop is closed. The upper diagram plots the timeseries of the target output, the system output, which is the motor command \( m(t) \) in this task, and the base rotation \( \theta(t) \). The middle and lower diagrams plot the corresponding sensory timeseries \( s(t) \) in odd and even numbers, respectively. (b) Plot showing an example of the sensorimotor timeseries when the system is driven by the closed-loop control with the external perturbations in the same experimental condition as (a). The perturbation to the arm is provided at around timestep 2700 – 2850. (c) The external perturbation is provided by a manual mechanical disturbance to the arm. (d) The average system outputs for a single period of a square function driven with open loop providing the target output as input. The plots for \( \tau_{\text{square}} = 12, 16, 20 \) and 24 are overlaid. The black line shows the target values as a reference. The time is shifted to set the switching point to \( t = 0 \). (e) The average error (MSE) plot with respect to each \( \tau_{\text{square}} \). The error bars show the standard deviations. For (d,e), the average system output and MSE are calculated by using 50 cycles of oscillations in each condition. See the electronic supplementary material for details.
the system output shows a phase shift compared with the target output after the perturbation, it is generating a square function with a required length of $\tau_{\text{square}}$.

To evaluate the maximal length of $\tau_{\text{square}}$ of a square function that our system can embed, we investigated an average system output for one period of a square function by clamping the feedback loop from the system output and providing the target output as input for each $\tau_{\text{square}}$ (figure 4d and see the electronic supplementary material for details). If the system is driven by the closed-loop control, the error in the system output would propagate to the motor command through the feedback loop, which makes it difficult to evaluate the limitation of the system performance efficiently. In figure 4e, according to the increase of $\tau_{\text{square}}$, the average system output starts to deviate largely from the target output. By calculating the system error by means of the MSE in this setting, we found that the error grows immediately when $\tau_{\text{square}}$ becomes larger than 18 (figure 4e). Consistent with this result, we observed that when $\tau_{\text{square}}$ is more than 18, the system cannot embed a correct square function anymore, or it simply stops, continuously providing 0 or 1 as output. Thus, we can speculate that our system possesses enough memory to be exploited for embedding a square function up to a length of around $\tau_{\text{square}} = 18$.

### 3.3. Function emulation tasks

In this final task, we aim to quantitatively characterize the intrinsic computational capacity of our system, particularly focusing on its memory capacity. By providing a random binary sequence to the motor command as input, the system should perform function emulation tasks using the resulting sensory timeseries. Because our system is not an abstract computational unit but has physical and mechanical constraints, we need to define a certain duration of time for one input state or symbol. We call this duration of time $\tau_{\text{state}}$. We found that when a random binary sequence is provided as motor commands in the form of $\tau_{\text{state}} < 5$, the motor overheats and stops. Accordingly, we performed our experiments with $\tau_{\text{state}} \geq 5$. In addition, we introduced a different timescale for I/O, defined as $t'$, which takes one input symbol as a unit. This means that $t'$ is increased by increments of 1 for each $\tau_{\text{state}}$ timestep (see the electronic supplementary material, figure S4e, for details).

The first function we aim to emulate is one that provides a delayed version of the input, i.e. $I(t' - n)$ $(n = 1, 2, \ldots)$ (see the electronic supplementary material for details). This task enables the direct evaluation of whether the system contains memory traces of a past input within the current sensory values, and is frequently used to evaluate the memory capacity of dynamical systems [19–21]. For descriptive purposes, we call this the short-term memory task. The second function we aim to emulate is the N-bit parity checker. The output should provide 0 if $\sum_{d=0}^{n} I(t' - d)$ is an even number; otherwise, it should provide 1, with $n = 1, 2, \ldots$ (see the electronic supplementary material for details). Note that it is actually a ‘$(n + 1)$-bit parity checker’ in our case. According to the definition, the system needs the memory of input symbols to previous $n$ symbols within the system to emulate this function. In addition, this function is a nonlinear function, which maps the input to a linearly inseparable state [28]. Because we are adjusting only the static linear weights externally, we can evaluate whether the system contains memory and nonlinearity to be exploited. This task is also common in the evaluation of the computational capacity of dynamical systems [29,30]. Along with the definition of the input symbol, we also need to determine how to define a corresponding sensory timeseries. Let us assume that an input symbol was provided at timestep $t(= t' \tau_{\text{state}})$. As a result, the arm generates corresponding transient dynamics until the next input symbol is provided at timestep $(t' + 1) \tau_{\text{state}}$. We define sensory values at $(t' + 1) \tau_{\text{state}} - 1$ as corresponding values $s(t')$ for this input symbol, which is one timestep before the next input symbol is provided (see the electronic supplementary material, figure S4e, for details). By providing random binary input sequences to the system over several trials for each parameter $\tau_{\text{state}}$ and $n$, we collected the sensory timeseries used for training. In the evaluation, both target functions are simultaneously evaluated over a previously unseen random input sequence (see the electronic supplementary material, figure S4e, for details).

Examples of the system performance for the short-term memory task and the N-bit parity check task with $\tau_{\text{state}} = 5$ and 11, respectively, can be found in figure 5 and electronic supplementary material, video S4. The system output shows almost a perfect match with the target output when $n = 1$ and 2 in $\tau_{\text{state}} = 5$ for the short-term memory task (figure 5a) and in $\tau_{\text{state}} = 11$ for the N-bit parity check task (figure 5b). For both tasks, the performance gradually gets worse when the delay $n$ is increased. To evaluate the influence of the parameters of $\tau_{\text{state}}$ and $n$ on the system performance, we introduced a measure based on mutual information, $\text{MI}_{\Sigma}$, between the system output and the target output [29]. This measure evaluates the similarity between the system output and the target output and, in our experiment, can take the value of 1 as maximum and 0 as minimum. Additionally, we introduced a measure called ‘capacity’, which is a summation of $\text{MI}_{\Sigma}$ over the delays, expressed as $C = \sum_{n=1}^{n_{\text{max}}} \text{MI}_{\Sigma}$, where $n_{\text{max}}$ is set to 10 in this analysis (see the electronic supplementary material for details). This measure can evaluate the system’s performance over the delays, which can take 10 as maximum and 0 as minimum in our experiment.

Figure 6a,b shows the results of the average $\text{MI}_{\Sigma}$ for each $n$ value and the average capacity for each $\tau_{\text{state}}$ for each task (see the electronic supplementary material for details of the setting). For the short-term memory task, when $\tau_{\text{state}}$ is increased, the value of $\text{MI}_{\Sigma}$ suddenly drops when $n$ is larger than 2 (figure 6a, left). For the capacity, increasing $\tau_{\text{state}}$ results, first, in a gradual decrease and then in saturation at a constant value for $\tau_{\text{state}} > 11$ (figure 6a, right). This can be explained by the behaviour of the arm (see electronic supplementary material, video S4)—if the length of the input symbol is short, it is more likely that the current transient dynamics contains the trace of previously provided input symbols. Considering that the arm base takes about nine timesteps, we assume that an input symbol was provided at timestep $(t' + 1) \tau_{\text{state}}$. As a result, the arm generates corresponding transient dynamics until the next input symbol is provided at timestep $(t' + 1) \tau_{\text{state}}$. We define sensory values at $(t' + 1) \tau_{\text{state}} - 1$ as corresponding values $s(t')$ for this input symbol, which is one timestep before the next input symbol is provided (see the electronic supplementary material, figure S4e, for details). By providing random binary input sequences to the system over several trials for each parameter $\tau_{\text{state}}$ and $n$, we collected the sensory timeseries used for training. In the evaluation, both target functions are simultaneously evaluated over a previously unseen random input sequence (see the electronic supplementary material, figure S4e, for details).
Figure 5. Examples of the output timeseries for the function emulation tasks. (a) Plots showing the example of the performance in the short-term memory task with $\tau_{\text{state}} = 5$. (b) Plots showing the example of the performance in the $N$-bit parity check task with $\tau_{\text{state}} = 11$. The open squares show the target outputs and the filled squares show the system outputs, and the cases for $n = 1, 2, 3$ and 4 are shown. (Online version in colour.)

this model cannot perform this task at all, suggesting that the performance of our system is purely based on the body dynamics (figure 6a, right).

For the $N$-bit parity check task, even if $\tau_{\text{state}}$ is small ($\tau_{\text{state}} = 5$), when $n = 1$ (figure 6b, left), MI$_{\text{a}}$ shows a smaller value than when $\tau_{\text{state}}$ is larger ($\tau_{\text{state}} = 10$ and 20). When $\tau_{\text{state}}$ gets larger ($\tau_{\text{state}} = 10$), MI$_{\text{a}}$ starts to show the highest value when $n = 1$, and a moderately high value when $n = 2$. If we increase $\tau_{\text{state}}$ further ($\tau_{\text{state}} = 20$), MI$_{\text{a}}$ still shows the highest value when $n = 1$, but the value when $n = 2$ starts to decrease. This tendency reflects the results of the capacity (figure 6b, right). The capacity shows a peak around $\tau_{\text{state}} = 9, 10$ and 11. The low values of capacity in $\tau_{\text{state}}$, less than 9 and larger than 11 are because of the low values of MI$_{\text{a}}$ for $n = 1$ and $n = 2$, respectively. Additionally, in this task, the model with a readout directly attached to the input cannot perform the emulation at all (figure 6b, right). Considering that the $N$-bit parity check task requires not only memory, but also nonlinearity to perform, this result suggests that, even if the transient dynamics of the arm possesses a high memory capacity when $\tau_{\text{state}}$ is low, it does not contain sufficient nonlinearity to be exploited. This is interesting, because this result is not detectable simply by looking at the arm motion. Furthermore, the results show that the amount of computational capacity depends on the type of motion generated in the arm.

We have further characterized the computational power of our system by comparing its performance with a conventional ESN, which has the same I/O settings with the same training procedures for the readout weights, the same number of computational nodes (10 fully coupled nodes with one bias term), and the same length of training and evaluation datasets.

It has been shown that the computational performance of an ESN is up to the spectral radius of the reservoir connectivity matrix [11]. In each task, we varied the spectral radius of the ESN from 0.05 to 2.0 and calculated the averaged capacity over 30 trials in each spectral radius value, with a new ESN in each trial. For the short-term memory task, the best capacity value of the ESN was 4.59 $\pm$ 0.38 (electronic supplementary material, figure S5, left), whereas our system showed the best value of 2.50 $\pm$ 0.07 when $\tau_{\text{state}} = 5$ (figure 6a, right), which was lower than the ESN. For the $N$-bit parity check task, the best capacity value of the ESN was 4.59 $\pm$ 0.38 (electronic supplementary material, figure S5, right), and the best capacity value of our system showed a similar value of 1.65 $\pm$ 0.07 when $\tau_{\text{state}} = 11$ (figure 6b, right).

Considering that soft bodies have multifaceted usages and advantages in addition to the computational abilities presented here, whereas the ESN is focused only on computational tasks, we think that our system performance is at a satisfactory level. Further details of these comparisons are given in the electronic supplementary material.

4. Discussion

In this study, we have systematically demonstrated that the body dynamics of the soft silicone arm can be exploited as computational resources. In particular, for the closed-loop control task, our results suggest that soft body dynamics...
can be sufficient to perform the task to control the body without the need of an external controller for additional memory capacity. This can be, for example, directly applied to the recently proposed octopus-inspired swimming robot [27] to generate the arm motion in a closed-loop manner exploiting the body dynamics itself, which largely outsources the computational load required to generate the motor command to the body. The technique presented here can be potentially applied to a wide class of soft robots, because the main component required is the soft body itself. Consequently, different types of morphology and material properties of robots that increase the computational capacity of the body should be explored in the future. In addition, developments in new types of sensors, which can effectively monitor body dynamics, would make the presented approach usable in additional applications. To conclude, we believe that we have presented a crucial step towards a novel control scheme for soft robots.

In reservoir computing studies, it has been established that to have powerful computational capabilities, a reservoir should have the properties of input separability and fading memory [10]. Input separability is usually achieved by a nonlinear mapping of the low-dimensional input to a high-dimensional state space. Fading memory is a property to uphold the influence of a recent input sequence within the system, which permits integration of stimulus information over time. This guarantees reproducible computation, for which the recent history of the signal is important. Our insight here was to exploit soft body dynamics as a reservoir. Passive body dynamics of soft materials typically tend to underactuated systems [7]. This naturally maps the actuation signal into the higher dimension of the soft body, which realizes the separability of the actuation signal. Furthermore, the interaction between the body and the environment (in our case, the underwater environment) implements fading memory, which takes a certain duration of time to relax when actuated due to the damping effect provided by the environment. Mechanical structures exhibiting these properties can also be exploited with our approach.

The framework presented in this study may also shed light on the role of the body in biological systems. Such systems have soft bodies that can adapt and behave effectively in a given ecological niche. For example, the octopus does not have any rigid components in its body but it shows extremely sophisticated behaviour that capitalizes on its body morphology and muscle structures [31]. In particular, we have

![Figure 6. The average value of $M(n)$ according to $n$ (left) and $C$ according to $\tau_{state}$ (right). (a) Plots showing the case for the short-term memory task. (b) Plots showing the case for the $N$-bit parity check task. Note that the capacities in the short-term memory task and the $N$-bit parity check task are expressed as $C_{\text{memory}}$ and $C_{\text{parity}}$ respectively. For each plot of $M(n)$, the cases for $\tau = 5, 10$ and 20 are shown. For each plot of $C$, the results of a logistic regression model (LR) that has a readout directly attached to the input are plotted as comparisons. Results show an almost 0 value for each $\tau_{state}$. The error bars show the standard deviations for each plot. (Online version in colour.)](http://rsif.royalsocietypublishing.org/)

rsif.royalsocietypublishing.org J. R. Soc. Interface 11: 20140437

![Figure 6. The average value of $M(n)$ according to $n$ (left) and $C$ according to $\tau_{state}$ (right). (a) Plots showing the case for the short-term memory task. (b) Plots showing the case for the $N$-bit parity check task. Note that the capacities in the short-term memory task and the $N$-bit parity check task are expressed as $C_{\text{memory}}$ and $C_{\text{parity}}$, respectively. For each plot of $M(n)$, the cases for $\tau = 5, 10$ and 20 are shown. For each plot of $C$, the results of a logistic regression model (LR) that has a readout directly attached to the input are plotted as comparisons. Results show an almost 0 value for each $\tau_{state}$. The error bars show the standard deviations for each plot. (Online version in colour.)](http://rsif.royalsocietypublishing.org/)
shown that a form of short-term memory, which is thought to be a functionality of the brain, can also be found in soft body dynamics. We think this line of studies is an interesting research direction to be explored further.

Funding statement. This work was partially supported by the European Commission in the ICT-FET OCTOPUS Integrating Project (EU project no. FP7-231608), and by the JSPS Postdoctoral Fellowships for Research Abroad.

References