Dynamically writing coupled memories using a reinforcement learning agent, meeting physical bounds

Théo Jules, Laura Michel, Adèle Douin, Frédéric Lechenault

Dynamically writing coupled memories using a reinforcement learning agent, meeting physical bounds.

Théo Jules,[1,*] Laura Michel,[2,*] Adèle Douin,[2] and Frédéric Lechenault[2]

[1] *Raymond and Beverly Sackler School of Physics and Astronomy, Tel Aviv University, Ramat Aviv, Tel Aviv, 69978, Israel*

[2] *Laboratoire de Physique de l'École Normale Supérieure, ENS, PSL Research University, CNRS, Sorbonne University, Université Paris Diderot, Sorbonne Paris Cité, 75005 Paris, France*

(Dated: September 12, 2022)

Bits manipulation in traditional memory writing is commonly done through quasi-static operations. While simple to model, this method is known to reduce memory capacity. We demonstrate how a reinforcement learning agent can exploit the dynamical response of a simple multi-bit mechanical system to restore its memory. To do so, we introduce a model framework consisting of a chain of bi-stable springs manipulated on one end by the external action of the agent. We show that the agent learns how to reach all available states for three springs, even though some states are not reachable through adiabatic manipulation, and that training is significantly improved using transfer learning techniques. Interestingly, the agent also points to an optimal system design by taking advantage of the underlying physics. Indeed, the control time exhibits a non-monotonic dependence on the internal dissipation, reaching a minimum at a cross-over shown to verify a mechanically motivated scaling relation.

INTRODUCTION

At first sight, memory seems like a fragile property of carefully crafted devices. However, upon closer inspection, various forms of information retention are present in a wide array of disordered systems [1]. Surprisingly, while they are governed by a very rugged and complex energy landscape, different disordered systems display similar history-dependent dynamical [2–5] and static [6, 7] responses. These observations led to a recent surge of interest in the Preisach model of hysteresis [8, 9]. The Preisach model views hysteresis cycles as two-state systems, an embryonic form of non-volatile memory. Note that other forms of memory with a different underlying structure exist [10, 11]. Combining these bistable elements in larger structures generates multi-stable systems with a memory capacity scaling exponentially with the number of elements. The simplicity of the model makes it applicable in many areas of physics including photonic devices [12], glassy [13], plastic [14, 15] and granular [16, 17] systems, spin ice [18], cellular automata [19] or even crumpled sheets [20] and origami bellows [21, 22]. More importantly, the Preisach model captures some remarkable features of complex real-life systems, such as return point memory [16, 17, 23–25]. Furthermore, the model allows for information to be written and read from the underlying system, making the internal memory mechanism adequate to store data. For this purpose, however, the Preisach model presents a major limitation: it stays grounded in the quasi-static framework of adiabatic transformations. As a result, the specific characteristics of each hysteresis cycle [26] or the addition of internal coupling [20, 22, 27] can considerably shrink the set of reachable stable states and effectively reduce the memory capacity of the system. In this paper, we show that a controller, in the form of a reinforcement learning agent, can go beyond this limitation and take advantage of the *dynamics* to reach all stable states, including adiabatically inaccessible states, effectively restoring the memory of the system to its full capacity. We base our study on a model framework akin to that introduced in [22, 28], i.e., a chain of bi-stable springs with three coupled units and internal dissipation controlled by a force applied on the last mass. Positional actuation would result in non-local interactions, which are known to significantly alter the memory structure of the system [29]. After successfully training the agent on a specified set of physical parameters, we demonstrate that transfer learning [30, 31] accelerates the training on different parameters and extends the region of parameter space that leads to learning convergence. Finally, we investigate the change of the dynamical protocol proposed by the trained control process for a single transition between two states as a function of the dissipation's amplitude. The transition duration presents a minimum for a critical value of the dissipation that appears to verify a physically motivated scaling relation, pointing to the fact that the agent learns how to harness the physics of the system to its advantage.

RESULTS

Model for a chain of bi-stable springs

We consider a one-dimensional multi-stable mechanical system composed of n identical masses m connected by

* These authors contributed equally to this work
Contact: theo.jules.physics@gmail.com

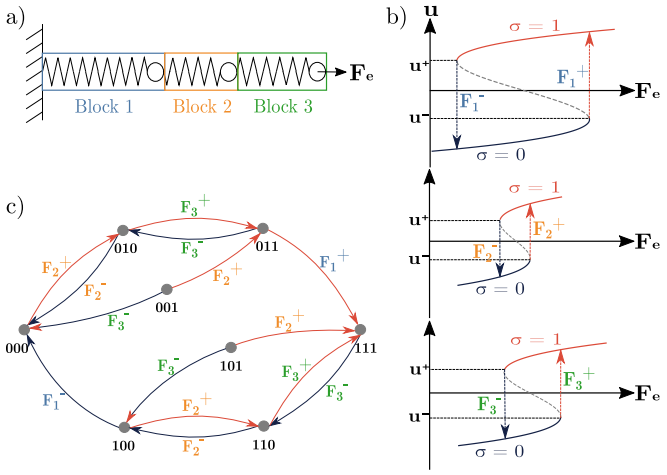

FIG. 1: Model for a chain of three coupled bi-stable spring-mass units (see *Methods* for more details on the parameters.). a) Schematics view of the model. The first unit is attached to a fixed wall and an external force F_e is applied to the last one. b) Deformation u of all three bi-stable springs under external load F_e. The switching fields $F_i^{\pm}$ are defined through Eq. (4). c) Transition graph where the nodes represent the stable configurations and the arrows the quasi-statically achievable transitions between them. The states 001 and 101 are "Garden of Eden" states (see main text) with this choice of disorder.

bi-stable springs in series, as shown in Fig. 1 a). For each spring i, we set a reference length l_i such that the deformation u_i of the spring reads

$$u_i(t) = x_i(t) - x_{i-1}(t) - l_i, \tag{1}$$

where $x_i(t)$ is the position of the i-th mass at time t. The tunable bi-stability of each spring is achieved through a generic quartic potential. We thus obtain the cubic force equations

$$F_i(u_i) = -ku_i(u_i + \delta_i^{(0)})(u_i - \delta_i^{(1)}), \tag{2}$$

with $k > 0$ is the stiffness of the spring, $\delta_i^{(0)} > 0$ and $\delta_i^{(1)} > 0$ correspond to meta stable configurations 0 and 1, and $u_i = 0$ corresponds to an unstable equilibrium. Other works on similar systems also considered trilinear forms [14, 28]. Still, our choice generates a smooth mechanical response and keeps its amplitude moderate for low deformation. This behavior plays a crucial role in the scaling analysis that we will present later. The first spring of the chain is attached to a fixed wall, a condition that imposes $x_0 = 0$, and we apply an external force F_e to the last mass. Finally, we consider that the system is bathed in an environmental fluid, resulting in a viscous force $F_{f,i}$ being exerted on each mass such that

$$F_{f,i} = -\eta \dot{x}_i, \tag{3}$$

where the dot indicates the derivative with respect to time $\dot{x} = \frac{dx}{dt}$ and η is a viscous coefficient.

Let us first consider the case $n = 1$ where the system is a single bi-stable spring attached to a mass. The solutions for mechanical equilibrium, schematized for three different springs in Fig. 1 b), displays two critical amplitudes $F^{\pm}$ such that

$$F^{\pm} = -F(u_{\mp}) \tag{4}$$

$$u_{\pm} = \frac{1}{3}\left(\delta^{(1)} - \delta^{(0)} \pm \sqrt{(\delta^{(0)})^2 + (\delta^{(1)})^2 + \delta^{(0)}\delta^{(1)}}\right) \tag{5}$$

where $u_{\pm}$ are both solutions of $\frac{\mathrm{d}F}{\mathrm{d}u} = 0$. Note that $F^+ > 0 > F^-$ since we defined both $\delta^{(0)}$ and $\delta^{(1)}$ to be positive. These two forces are essential to describe the stability of the system. Indeed, we observe two branches σ of stable configurations, that we call `0` (for $u < 0$) and `1` (for $u > 0$). Both branches present a solution for the mechanical equilibrium $F(u) + F_e = 0$ when F_e remains in the range $[F^-, F^+]$: the system has two stable states. However, as soon as F_e gets beyond this range, either the branch $\sigma =$ `0` (for $F_e > F^+$) or $\sigma =$ `1` (for $F_e < F^-$) vanishes.

These dynamical properties lead to an hysteresis cycle that can be described with a simple experiment. We start with the spring at rest with $\sigma = 0$, a configuration which corresponds to $u = -\delta^{(0)}$, and slowly pull on its free end, increasing F_e while keeping mechanical equilibrium. As a response, the system slightly stretches and u increases. This continues until $F_e = F^+$ where the branch $\sigma =$ `0` disappears. At this point, if we continue to increase F_e, the system necessarily jumps to the branch $\sigma =$ `1`, with $u > 0$. If we now decrease F_e, u decreases accordingly until $F_e = F^-$ where the system has to jump back to the original branch $\sigma =$ `0`. If we only consider the stability branches, the described hysteresis cycle corresponds to a so-called hysteron, the basic block of the Preisach model [8]. In a Preisach model, a set of n independent hysterons is actuated through an external field. Each hysteron has two states, `0` and `1`, and two switching fields, F_i^+ and F_i^-, that characterize when each hysteron σ_i switches between states. As long as the switching fields are unique and the hysterons are independent, *i.e.* the switching fields do not change with the state of the system, and the Preisach model is able to predict the possible transitions between the 2^n configurations. Our system, n bi-stable spring/mass in series under quasi-static actuation, fulfills the local mechanical equilibrium and independence assumptions.

To get a better visual representation of the model's predictions, the quasi-statically achievable transitions between states are modeled as the edges of a directed graph, where the nodes represent the stable configurations. The allowed transitions correspond to the switch of a single hysteron. The exact topology of the transition diagram exclusively depends on the relative values of the switching fields [26]. If the order

of the positive and negative switching switching fields are opposite, i.e. $0 < F_i^+ < F_j^+$ and $0 > F_i^- > F_j^-$ for all $i < j$, every state is reachable from any other state. Otherwise, there exists isolated configurations which can never be reached again once left. As is customary, we call these unreachable configurations "Garden of Eden" (GoE) states (see [22] for a historical account). The number of GoE states depends on the permutation in the order of the switching fields [26]. In real systems, these permutations might come from defects in the conception of the bistable units with close switching fields, for instance. An illustration for the case $n = 3$ is given in Fig. 1 c) where $F_1^- < F_2^- < F_3^-$ and $F_2^+ < F_3^+ < F_1^+$. In this configuration, both `001` and `101` are GoE states. The existence of GoE states severely limits the total number of reachable states, and as a result, the amount of information that can be stored in the system.

A solution to overcome this limitation is to break the quasi-static assumption and to take advantage of the *dynamics* as a means to reach GoE configurations. However, this approach foregoes a major upside of the Preisach model, its simplicity. Indeed, applying Newton's second law of motion to each mass yields a description of the dynamical evolution for the system through a system of n coupled non-linear differential equations

$$\begin{cases} m\ddot{x}_i = F_i(u_i) - F_{i+1}(u_{i+1}) + F_{f,i} \quad \text{for } 0 < i < n, \\ m\ddot{x}_n = F_n(u_n) + F_e(t) + F_{f,n}. \end{cases} \tag{6}$$

While providing a clear experimental protocol to make the system change configuration is straightforward in the quasi-static regime, the non-linear response of the springs and the coupling between the equations make a similar analysis a hefty challenge in the dynamical case. In particular, other methods of dynamical control like linear quadratic regulators do not manage to properly handle the transitions to GoE state for all physical parameters and initial conditions (see *Supplemental Materials*). In the following, we demonstrate how the use of artificial neural networks and reinforcement learning unlocks this feat.

Reinforcement learning to control the dynamics

Reinforcement Learning (RL) is a computational paradigm that consists in optimizing, through trial-and-error, the actions of an agent that interacts with an environment. RL has been shown to be an effective method to control multistable systems with nonlinear dynamics [32–35]. In RL, the optimization aims to maximize the cumulative reward associated with the accomplishment of a given task. At each step t, the environment is described by an observable state s_t. The agent uses this information, in combination with a policy μ, to decide the action a_t to be taken, *i.e.* $\mu(s_t) = a_t$. This action brings the environment to a new state s_{t+1}, and grants the agent with a reward r_t quantifying its success with respect to the final objective. An *episode* ends when that goal is reached or, if not, after a finite time. The goal of training is to learn a policy that maximizes the agent's cumulative reward over an episode. When the control space is continuous, one can resort to an *actor-critic* architecture [36], which is based on two Artificial Neural Networks (ANN) learning in tandem. One network, called the actor, generates a sensory-motor representation of the problem in the form of a mapping of its parameter space θ into the space of policies, such that $\mu = \mu_\theta$. This setup limits the type of policy considered and makes the search for an optimal policy computationally tractable. The optimization of the actor necessitates an estimation of the expected reward at long time. This is the role of the second ANN, the critic, which learns to evaluate the decisions of the actor, and how it should adjust them. This is done through the same bootstrapping of the Bellman's equation as that used in Q-learning [37]. The successive trials - resulting in multiple episodes - are stacked in a finite memory queue (FIFO), or replay buffer, and after each trial the ANN tandem is trained on that buffer, thus progressively improving their decision policy and the quality of the memory. After testing multiple RL algorithms (see *Supplementary Materials*), we found that an architecture based on the Twin Delayed Deep Deterministic Policy Gradient (TD3) [38] worked best for the task at hand.

In our system, the environment consists in the positions and velocities of the masses, an action is a choice for the values of the force applied to the last mass in the chain at time t, and the goal is to bring the system close to a given meta-stable memory state in a given time $t_{\max}$ - close enough that it cannot switch states if let free to evolve. At the start of an episode, the environment is randomly initialized, and a random target state is set. The information provided as an input to the networks includes the position x and velocity $\dot{x}$ of all the masses in addition to the one-hot encoded target configuration. Then the policy decides on the force F_e applied to reach the next step. F_e is taken from a predefined interval $[-F_{\max}, F_{\max}]$. Here, we set $F_{\max}$ to 1 N. With this external force, the dynamical evolution of the system for a single time step is simulated by solving the differential equations (6) numerically with a Runge-Kutta method of order 4. The reward from this action is computed relative to the newly reached state: we give a penalty ($r_t < 0$) with an amplitude proportional to the velocity of the masses and to the distance of the masses from their target rest positions (see equation (12)). After every step, the replay buffer is updated with the corresponding data. The critic is optimized with a batch of data every step, while the actor is optimized every two steps. The episode stops if the system is sufficiently close to rest in the correct configuration, in which case a large positive reward ($r_t \gg 1$) is granted, or after $t_{\max}$. Then a new episode is started, and the algorithm is repeated for

a predefined number of episodes. More details on the learning protocol are available in *Methods*.

Using the described method, we trained our ANNs on a chain of three bi-stable springs specifically designed to display GoE states, as detailed in *Methods* and illustrated in Fig. 1 c. Interestingly, the networks achieve a 100% success rate at reaching any target state - including the GoE states - in less than 10 000 episodes, as shown in Fig. 2. a). This approach remains successful even if we only keep the first two decimals of the environment states, a promising behavior for real-life applications (see Supplementary Materials). We thus accomplished our initial objective and designed a reliable method that produces protocols for the transitions to any stable configurations, restoring the memory of the device to its full capacity.

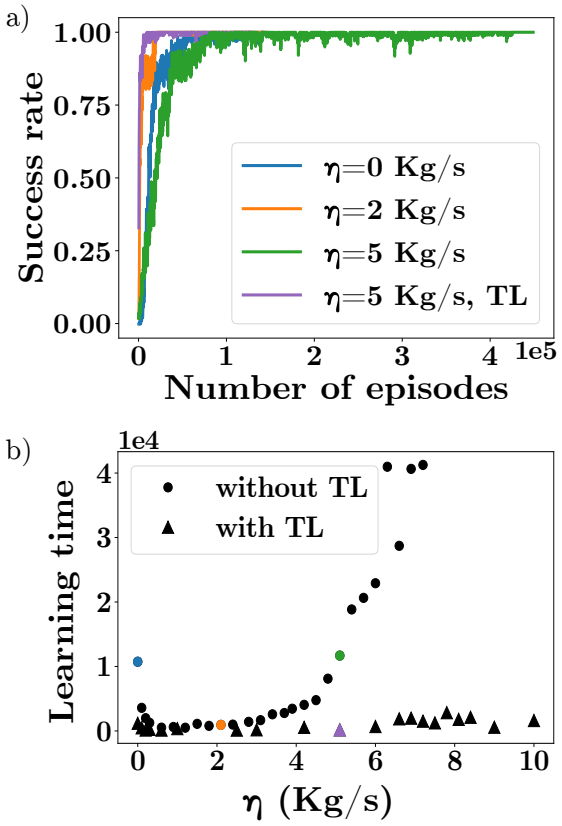

FIG. 2: Training dynamics of the RL agent on the model presented in Fig. 1 and different values of the viscous coefficient. a) Evolution of the success rate over a sliding span of 100 episodes during training. For the blue, green, and orange curves, the ANN was initialized randomly. For the purple curve, the ANN was initialized using the weights of a previously model trained with η=4 Kg/s. b) Learning time against the viscous coefficient with and without TL.

In order to evaluate the robustness of the decision making process of the policy and gain insights on the mechanisms involved, we study how the physical parameters of the system influence the designed solutions. We chose to focus on a quantity that deeply affects the dynamics of the masses and has simple qualitative interpretation: the viscous coefficient η. To observe its effect on the designed policy, we trained agents with random initialization of weights for a range of η while keeping all other physical parameters fixed. The learning time, defined as the number of episodes before the success rate reaches 80% during training, is shown in Fig. 2 b) as a function of η. Even though the algorithm manages to learn the transitions for a wide range of η, the learning time varies significantly. Notably, the learning time gets longer for very low viscous coefficients (here $\eta < 0.1$ kg/s) but also seems to diverge at very high η.

Due to the continuous nature of the system, we expect minor modifications of the physical parameters to not catastrophically change the dynamics. With this assumption, we employed Transfer Learning (TL) techniques [30, 31, 39] to accelerate the learning phase. We initialized the ANNs with the weights of ANNs already trained on a similar physical system instead of random weights. The expectation is that some physical principles learned during training remain applicable for solving the new problem. Thus, the transfer of weights initializes the networks closer to a good solution. Please note this assumption might not hold if the change of parameters leads to the appearance of a significantly different solution [39], for instance in the presence of bifurcations. In our system, TL results in quicker learning. In Fig. 2. b), we compare the learning time, defined as the number of episodes it takes for the success rate to reach 0.8 for the first time, for networks initialized with and without TL. For networks trained with TL, we slowly increased η from 2 kg/s up to 10 kg/s and decreased it from 2 kg/s to 0 kg/s, transferring the learned weights at each increment. TL effectively divides by up to 30 the learning time for very high viscosity and allows to reach otherwise non-converging regions. We also noticed that smaller increments of the parameters lead to faster learning (see Supplementary Materials). Consequently, we can use finer discretizations to explore the physical parameters while keeping computation time reasonable.

By mixing RL and TL, we generated an algorithm that quickly produces precise transition protocols to any stable state for chains of bi-stable springs, including GoE states. In the next section, we analyze the properties of the force signals produced by the ANN and investigate how they relate to the dynamics of the system as the viscous coefficient η is varied.

How damping affects the control strategy

The intensity of the damping impacts the dynamical response of the system, which significantly affects the actuation protocol proposed by the ANN. To study

these variations, we selected a unique transition (111 → 001), recorded the signal of the force generated by the agent, and computed the corresponding mechanical energy injected into the system at each time step for different values of η, as shown in Fig. 3. Please note that the state 001 is a GoE state. We observe that the relation between η and the time it takes to reach the target state is non-monotonic. We identify the minimum episode duration, corresponding to the most efficient actuation, with a critical viscous coefficient $\eta = \eta_c$. Interestingly, this minimum also marks the transition between two qualitatively different behaviors of the control force: a high-viscosity regime ($\eta > \eta_c$) and a low-viscosity regime ($\eta < \eta_c$) as shown in Fig. 3 c) and d). In the high-viscosity regime, F_e always saturates its limit value, and only changes sign a few times per episode. Remarkably, each sign change occurs roughly when a mass is placed at the correct position. In contrast, the force signal in the low-viscosity regime appears less structured, with large fluctuations between consecutive steps.

In order to qualitatively explain these different behaviors, we focus our analysis on the energy transfer between the external controller and the system. The starting and the final states are stable configurations at rest. Consequently, they both correspond to local energy minima and the agent has to provide mechanical energy to the system in an effort to overcome the energy barriers between these configurations. After crossing the barriers, the surplus of kinetic energy has to be removed to slow down the masses and trap them in the well associated with the targeted minimum. In the low-viscosity regime, the internal energy dissipated due to viscosity is small. As a result, the protocols require phases where the agent is actively draining energy from the system. After a short initial phase of a few steps, where much energy is introduced into the system by setting the external load to its maximal value, a substantial fraction of the remainder of the episode involves careful adjustments to remove the kinetic energy. We expect this precise control of the system's internal energy to be the underlying reason behind the increase of the learning time at low η. In the high-viscosity regime on the other hand, the viscosity is able to dissipate the extra energy without further intervention. However, it also slows the dynamic, effectively translating into increased episode duration.

While we established the characteristics of the designed protocols in both regimes, we have yet to define a quantitative estimation of the crossover between regimes.

Drawing inspiration from these protocols, we consider a simplified situation where the external force saturates the constraint $F_e(t) = F_{\max}$, and where the switching fields are very small, i.e. $F_{\max} \gg |F_i^{\pm}|$, $i = 1, 2, 3$.

Since all the masses start in a stable equilibrium, the mechanical response of the chain to the external load is very soft, at least for small enough displacements at the start of the episode.

Thus, we can approximates the dynamics of the last

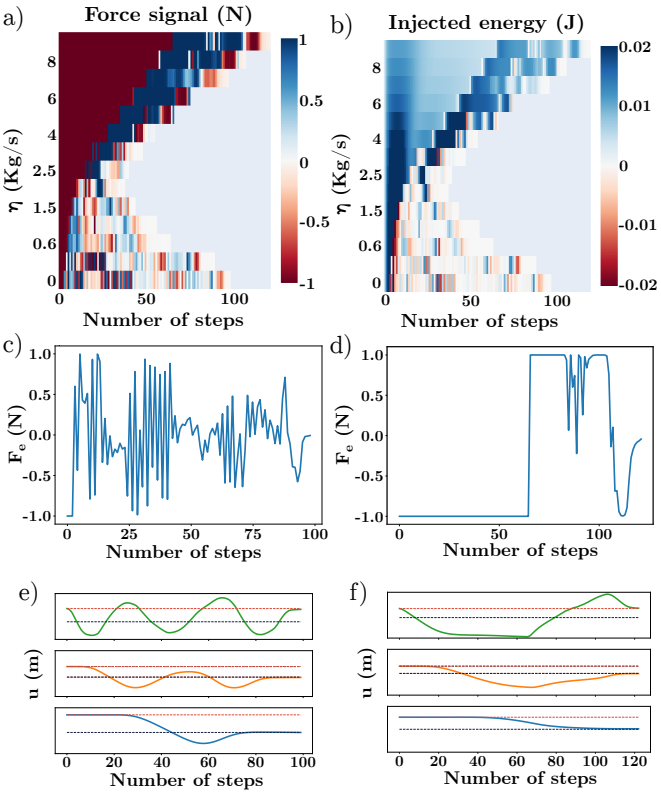

FIG. 3: Analysis of the protocol proposed by the RL agent for the transition 111 → 001 with different values of the viscous coefficient. a) Force signal and b) injected energy during the transition. The injected energy is computed by multiplying the chain's elongation with the value of the external force. c) Force signal and e) deformation of each bi-stable spring during the transition for $\eta = 0$ kg/s (low-viscosity regime). The colors blue, orange, and green correspond to the deformation of the first, second, and third spring, respectively. The subplots for the deformations all have the same height with edge values [-0.07, 0.19] meter. The dashed lines represent the stable equilibria $\delta^{(0)}$ (black) and $\delta^{(1)}$ (red). d) Force signal and f) deformation of each bi-stable spring during the transition for $\eta = 9$ kg/s (high-viscosity regime). The subplots for the deformations all have the same height with edge values [-0.14, 0.24] meter.

mass by the ordinary differential equation

$$\tau \ddot{x}_3 \approx v_{\max} - \dot{x}_3, \tag{7}$$

which solves into

$$\dot{x}_3(t) = v_{\max}(1 - e^{-t/\tau}). \tag{8}$$

with a relaxation time $\tau = \frac{m}{\eta}$ and a saturation velocity $v_{\max} = \frac{F_{\max}}{\eta}$. The relaxation time τ corresponds to the time it takes for dissipation to take over inertia. This transition is also associated to a length scale L_η such

that

$$L_\eta = \tau v_{\max} = \frac{mF_{\max}}{\eta^2} \tag{9}$$

Let us now define more precisely what we mean by small displacement. With our assumptions, and due to the asymptotic shape of the potential, the typical relative distance L_e for which the mechanical response becomes of the same magnitude as the external load verifies

$$kL_e^3 \sim F_{\max} \tag{10}$$

It is thus clear that if $L_\eta \ll L_e$, the system will be dominated by dissipation and converge to equilibrium without further oscillations. On the other hand, if $L_\eta \gg L_e$, neighboring masses will rapidly feel differential forces and inertia will dominate. Interestingly, equating these two length scales allows to point to a critical dissipation at the frontier of these two regimes

$$\eta_c \sim m^{1/2} k^{1/6} F_{\max}^{1/3} \tag{11}$$

To test this prediction, we investigate how the damping crossover η_c observed in designed policies varies as the masses m and the maximum force $F_{\max}$ are varied, exploring more than two orders of magnitude for both parameters. As show in Fig.4, the results present an excellent agreement with the proposed scaling argument [11].

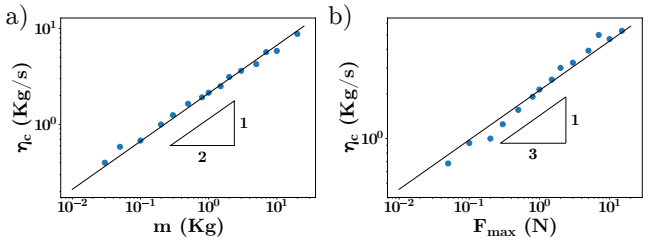

FIG. 4: Evolution of the critical viscous coefficient η_c with respect to a) the masses m and b) the maximum amplitude of the external load $F_{\max}$. The lines show the scaling behavior $\eta_c \propto m^{1/2} F_{\max}^{1/3}$.

DISCUSSIONS

We have shown a proof of concept of general memory writing operations in a strongly non-linear system of coupled bi-stable springs by a reinforcement learning agent. In particular, we found that this technique allows reaching otherwise unreachable memory states dynamically. Interestingly, the agent appears to learn how to harness the physics underlying the behavior of the system: its control strategy changes qualitatively as the viscous coefficient is varied, from a relatively simple actuation in the large dissipation regime to a jerky dynamical behavior aimed at extracting the excess energy in the small dissipation regime, two significantly different modes of control. This transition coincides with a change in the system's internal response, from an over-damped to an inertial response. In that sense, the networks were able to gather and share with the authors some insightful knowledge about the physics of the memory system, thus displaying some form of intelligence in *understanding* the challenges it was asked to tackle, though it is unclear at this point whether this result is specific to this controller. Key stakes of future works will consist in identifying the cognitive structures established by the agent to complete the learned tasks, i.e. by rationalizing its neural activity and learning dynamics, using this knowledge to learn transitions for a higher number of coupled units. Indeed, while we managed to successfully train networks on a system of up to six springs, systems with more than four springs required a significant change in the architecture of the neural network (see *Supplement Materials*). Another promising path is to look into optimizing the control signal. Here, we present how the ANNs found *one* method to reach GoE states. We expect further tweaking of the reward, for instance by penalizing longer episodes or higher energy transfer between the agent and the system, to lead to better control patterns. Finally, our study is anchored on the Preisach model, a generic model applicable to a large variety of physical systems. We believe the reinforcement learning method to reach GoE states remains relevant in many real-life situations. We leave the confirmation to future studies.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

CODE AVAILABILITY

The code used to produce the results of this study is available online [40].

ACKNOLEDGEMENTS

Research by T.J. was supported in part by the Raymond and Beverly Sackler Post-Doctoral Scholarship.

AUTHOR CONTRIBUTIONS

T.J., F.L. and A.D designed research; T.J., L.M and A.D. wrote algorithms; L.M. produced and analyzed

data; T.J. produced scaling analysis; T.J., L.M. and F.L. wrote the paper.

COMPETING INTERESTS

The authors declare no competing interests.

[1] N. C. Keim, J. D. Paulsen, Z. Zeravcic, S. Sastry, and S. R. Nagel, Reviews of Modern Physics **91**, 035002.
[2] A. Kovacs, Adv. Polym. Sci **3**, 394 (1963).
[3] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, Nature **453**, 80 (2018).
[4] A. Prados and E. Trizac, Physical Review Letters **112**, 198001.
[5] T. Jules, F. Lechenault, and M. Adda-Bedia, Physical Review E **102**, 033005.
[6] K. Matan, R. B. Williams, T. A. Witten, and S. R. Nagel, Phys. Rev. Lett. **88**, 076101 (2002).
[7] J. Diani, B. Fayolle, and P. Gilormini, European Polymer Journal **45**, 601 (2009).
[8] F. Preisach, Zeitschrift für Physik **94**, 277 (1935).
[9] I. D. Mayergoyz, Physical Review Letters **56**, 1518.
[10] Y. Abu-Mostafa and J. S. Jacques, IEEE Transactions on Information Theory **31**, 461.
[11] K. Deng, S. Zhu, G. Bao, J. Fu, and Z. Zeng, IEEE Transactions on Neural Networks and Learning Systems , 1 (2021).
[12] C. Valagiannopoulos, A. Sarsen, and A. Alu, IEEE Transactions on Antennas and Propagation **69**, 7720 (2021).
[13] C. W. Lindeman and S. R. Nagel, Science Advances **7**, eabg7133 (2021).
[14] G. Puglisi and L. Truskinovsky, Continuum Mechanics and Thermodynamics **14**, 437 (2002).
[15] I. Regev, I. Attia, K. Dahmen, S. Sastry, and M. Mungan, Physical Review E **103**, 062614.
[16] N. C. Keim, J. Hass, B. Kroger, and D. Wieker, Physical Review Research **2**, 012004 (2020).
[17] N. C. Keim and J. D. Paulsen, Science Advances **7**, 10.1126/sciadv.abg7685.
[18] A. Libál, C. Reichhardt, and C. O. Reichhardt, Physical Review E **86**, 021406 (2012).
[19] J. Goicoechea and J. Ortín, Physical review letters **72**, 2203 (1994).
[20] H. Bense and M. van Hecke, Proceedings of the National Academy of Sciences **118** (2021).
[21] H. Yasuda, T. Tachi, M. Lee, and J. Yang, Nature communications **8**, 1 (2017).
[22] T. Jules, A. Reid, K. E. Daniels, M. Mungan, and F. Lechenault, Physical Review Research **4**, 013128 (2022).
[23] J. A. Barker, D. E. Schreiber, B. G. Huthand, and D. H. Everett, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **386**, 251 (1983).
[24] J. M. Deutsch, A. Dhar, and O. Narayan, Phys. Rev. Lett. **92**, 227203 (2004).
[25] M. Mungan and M. M. Terzi, Annales Henri Poincaré **20**, 2819 (2019).
[26] M. M. Terzi and M. Mungan, Phys. Rev. E **102**, 012122 (2020).
[27] M. van Hecke, Phys. Rev. E **104**, 054608 (2021).
[28] G. Puglisi and L. Truskinovsky, Journal of the Mechanics and Physics of Solids **50**, 165 (2002).
[29] R. C. Rogers and L. Truskinovsky, Physica B: Condensed Matter **233**, 370 (1997).
[30] S. J. Pan and Q. Yang, IEEE Transactions on knowledge and data engineering **22**, 1345 (2009).
[31] M. E. Taylor and P. Stone, Journal of Machine Learning Research **10** (2009).
[32] S. Gadaleta and G. Dangelmayr, Chaos: An Interdisciplinary Journal of Nonlinear Science **9**, 775 (1999).
[33] S. Gadaleta and G. Dangelmayr, Physical Review E **63**, 036217 (2001).
[34] X.-S. Wang, J. D. Turner, and B. P. Mann, Journal of Vibration and Control **27**, 502 (2021).
[35] A. N. Pisarchik and U. Feudel, Physics Reports **540**, 167 (2014).
[36] V. Konda and J. Tsitsiklis, Advances in neural information processing systems **12** (1999).
[37] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **42**, 591 (2011).
[38] S. Fujimoto, H. Hoof, and D. Meger, in *International conference on machine learning* (PMLR, 2018) pp. 1587–1596.
[39] K. Weiss, T. M. Khoshgoftaar, and D. Wang, Journal of Big Data **3**, 10.1186/s40537-016-0043-6 (2016).
[40] L. Michel, T. Jules, and A. Douin, laura042/Multistable_memory_system: v0, `https://doi.org/10.5281/zenodo.6514157` (2022).
[41] Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa, Journal of Machine Learning Research **22**, 1 (2021).
[42] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym.
[43] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).

METHODS

Physical parameters of the three springs system

The physical parameters for the three springs system that we consider in Fig. 1, Fig. 2 and Fig. 3 are detailed in table I. All length are in meters. This choice leads to a single permutation in the order of the switching fields and the presence of two GoE states.

$\delta_1^{(0)}$	$\delta_1^{(1)}$	$\delta_2^{(0)}$	$\delta_2^{(1)}$	$\delta_3^{(0)}$	$\delta_3^{(1)}$ (m)	m (Kg)	k (N/m^3)
0.050	0.050	0.040	0.020	0.030	0.045	1	88.7

TABLE I: Physical parameters for the system with three springs.

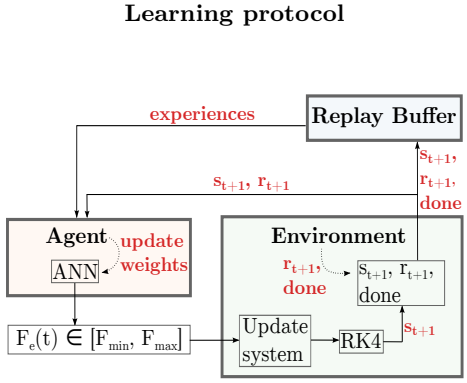

FIG. 5: Architecture of the learning protocol.

We used the TD3 agent implemented in [41] combined with Gym environments [42] to solve our control problem. We provide a schematic view of the learning protocol in Fig. 5 and we present more details about the TD3 algorithm and Gym environments in Supplementary Materials. The precise architecture of both the policy (Actor) and the Q-functions (Critics) are summarized in Fig 6 and table II. We start the training with random initial policy and Q-functions parameters. The weights and biases of each of layer are sampled from $U(-\sqrt{l}, \sqrt{l})$ where U is the uniform distribution and $l = \frac{1}{in\_features\_l}$, $in\_features\_l$ being the size of the input of the layer.

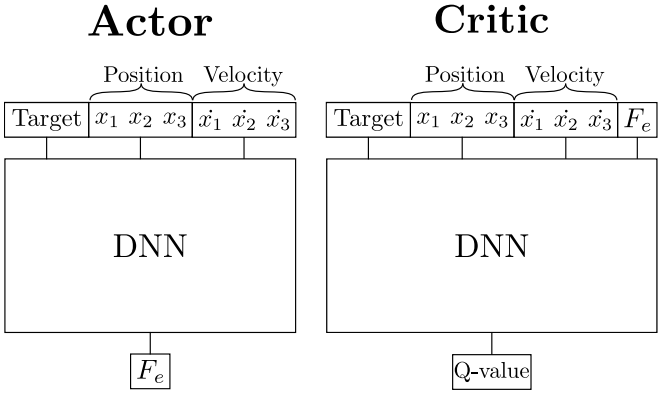

FIG. 6: Architecture of the actor and critic. "DNN" corresponds to the neural networks detailed in table II.

The task of the agent involves reaching a stable configuration close to rest, starting from random initial conditions. At the beginning of an episode, a target state is randomly chosen and the initial positions and velocities are randomly sampled from the respective intervals [$\delta_i^{(0)}$ - 0.2, $\delta_i^{(1)}$ + 0.2] and [-0.1, 0.1]. For the first 10 000 steps, actions are sampled uniformly from [-$F_{\max}$, $F_{\max}$] without concerting the policy or Q-functions. Once this exploratory phase is completed, the agent starts following the policy generated by the ANNs. At each time step t, the agent observes the current state of the system s_t (composed of the position, the velocity and the target state of each mass) and chooses a force $F_e(t)$ in the interval [-$F_{\max}$, $F_{\max}$] in consequence. The selected force, to which is added a noise taken from a Gaussian distribution of mean 0 and standard deviation 0.1, brings the system to a new state s_{t+1} computed by Runge-Kutta method of order 4. The parameters controlling the added noise were taken directly from [41] without further optimizations. At each time step (constant $F_e(t)$), the RK4 method is done through 10 successive iterations for a total duration of 0.1 s. Each of those iterations changes the state of the system. Once the numerical resolution is completed, the agent receives a reward r_t given by function (12)

Policy layer	Policy activation
Linear (400)	Relu
Linear (300)	Relu
Linear (1)	Tanh
Q-function layer	**Q-function activation**
Linear (400)	Relu
Linear (300)	Relu
Linear (1)	None

TABLE II: Policy (Actor) and Q-function (Critic) models.

$$r_t = -\sum_{i=1}^{N}(u_i^{(t+1)} - \delta_i^{(target)}) - \frac{1}{2}\sum_{i=1}^{N}\dot{x}_i^{(t+1)} \qquad (12)$$

where $target$ is a variable equal to 0 or 1, $u_i^{(t+1)}$ and $\dot{x}_i^{(t+1)}$ are respectively the displacement and velocity of the ith mass at time t+1. The parameters of the environment are summarized in the table III.

All the steps are stocked in the replay buffer, which possesses a finite maximal size of 1e6 steps. Each new step overwrites the oldest stored one when the buffer is full. This process allows for the continuous improvement of the available training dataset during training. The Q-functions and the policy are then updated. The Q-functions are updated at every step, while the policy is updated every two steps. Both the Q-functions and the policy are updated using the Adam algorithm [43] with a learning rate of 0.001 and a batch size of 100 experiences randomly sampled from the Replay Buffer. The hyperparameters for optimization are summarized in table IV. The operation goes on until either the agent reaches the goal, at which point it receives a reward of 50, or 200 steps are exceeded. At this stage, the environment is reset, giving place to a new episode. This algorithm is repeated for a predefined number of episodes.

Parameter	Value
$F_{\max}$	1 (N)
dt	0.1 (s)
n_res	10
max_episode_len	200
success_pos	0.005 (m)
success_vel	0.01 (m/s)
success_r	50
penalty_pos	1
penalty_vel	$\frac{1}{2}$

TABLE III: Environment parameters. dt : discretization time, n_res : number of iterations for the numerical resolution, max_episode_len : maximum number of steps per episodes, success_pos : success condition on the position, success_vel : success condition on the velocity, success_r : success reward, penalty_pos : penalty coefficient on the position, penalty_vel : penalty coefficient on the velocity.

Hyper-parameter	Value
Policy optimizer	Adam
Policy learning rate	0.001
Q-function optimizer	Adam
Q-function learning rate	0.001
γ	0.99
Replay buffer size	1e6
Exploration time	10 000 (steps)
Batch size	100
Policy update interval	2 (steps)
Q-function update interval	1 (step)

TABLE IV: TD3 agent hyperparameters.

Transfer Learning

We employed TL when changing only a single physical parameter for all of the numerical experiments presented in this paper. For the initial exploration presented in Fig. 3, we started with $F_{\max} = 1$ N, $m = 1$ kg and $\eta = 2$ kg/s. Then, we slowly explore the space of η with TL as detailed in *Results*. We used these new trained networks as the starting point of another TL, this time either changing the mass m of the maximum external force $F_{\max}$. These led to the scaling of η_c with respect to these parameters presented in Fig 4.