Chaos

Restoring chaos using deep reinforcement learning

Cite as: Chaos **30**, 031102 (2020); https://doi.org/10.1063/5.0002047
Submitted: 21 January 2020 . Accepted: 04 March 2020 . Published Online: 19 March 2020

Sumit Vashishtha, and Siddhartha Verma

View Online Export Citation CrossMark

ARTICLES YOU MAY BE INTERESTED IN

Amplification of explosive width in complex networks
Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 031101 (2020); https://doi.org/10.1063/5.0003410

Synchronization transition from chaos to limit cycle oscillations when a locally coupled chaotic oscillator grid is coupled globally to another chaotic oscillator
Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 033121 (2020); https://doi.org/10.1063/1.5134821

Supervised chaotic source separation by a tank of water
Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 021101 (2020); https://doi.org/10.1063/1.5142462

Restoring chaos using deep reinforcement learning

Cite as: Chaos **30**, 031102 (2020); doi: 10.1063/5.0002047
Submitted: 21 January 2020 · Accepted: 4 March 2020 ·
Published Online: 19 March 2020

Sumit Vashishtha[1,a)] and Siddhartha Verma[1,2,b)]

AFFILIATIONS

[1] Department of Ocean and Mechanical Engineering, Florida Atlantic University, Boca Raton, Florida 33431, USA
[2] Harbor Branch Oceanographic Institute, Florida Atlantic University, Fort Pierce, Florida 34946, USA

[a)] **Electronic mail:** vash.sumit@gmail.com
[b)] **Author to whom correspondence should be addressed:** vermas@fau.edu

ABSTRACT

A catastrophic bifurcation in non-linear dynamical systems, called crisis, often leads to their convergence to an undesirable non-chaotic state after some initial chaotic transients. Preventing such behavior has been quite challenging. We demonstrate that deep Reinforcement Learning (RL) is able to restore chaos in a transiently chaotic regime of the Lorenz system of equations. Without requiring any *a priori* knowledge of the underlying dynamics of the governing equations, the RL agent discovers an effective strategy for perturbing the parameters of the Lorenz system such that the chaotic trajectory is sustained. We analyze the agent's autonomous control-decisions and identify and implement a simple control-law that successfully restores chaos in the Lorenz system. Our results demonstrate the utility of using deep RL for controlling the occurrence of catastrophes in non-linear dynamical systems.

Numerous examples arise in fields ranging from mechanics to biology where the disappearance of chaos can be detrimental. Preventing such transient nature of chaos has proved to be quite challenging. In this paper, we demonstrate that Reinforcement Learning (RL), which is a specific class of machine-learning techniques, is capable of discovering effective control mechanisms in this regard. The autonomous control algorithm is able to prevent the disappearance of chaos in a specific non-linear system, without requiring any *a priori* knowledge about the underlying dynamics. We analyze the autonomous decisions taken by the algorithm to understand how the system's dynamics are impacted, which in turn allows us to formulate a simple control-law capable of restoring chaotic behavior. The reverse-engineering approach adopted underlines the immense potential of the techniques used here to discover effective control strategies in complex dynamical systems. We note that the autonomous nature of the learning algorithm makes it applicable to a diverse variety of non-linear systems, which highlights the potential of RL-enabled control for regulating other crisis-like catastrophic events.

I. INTRODUCTION

Chaos is desirable and advantageous in many situations. For instance, in mechanics, exciting the chaotic motion of several modes spreads energy over a wide frequency range,[1] thereby preventing undesirable resonance. Chaotic advection in fluids enhances mixing, as chaos brings about an exponential divergence of fluid packets that are initially in close proximity.[2] In biology, the absence of chaos may lead to an emergence of synchronous dynamics in the brain, which can result in epileptic seizures.[3] Moreover, the absence of chaos may also indicate the presence of other pathological conditions.[4,5]

In some cases, chaos can become transient in nature, where the dynamics eventually converge to non-chaotic attractors. The typical route by which this happens is known as a crisis,[6] where for certain parameter-values of the non-linear system, a chaotic-attractor collides with its basin-boundary and becomes a saddle. A saddle has a fractal structure with infinitely many gaps along its unstable-manifold. Any initial condition attracted toward this chaotic-attractor-turned-saddle escapes to an external periodic- or a fix-point-attractor. Such transient chaos is often undesirable and has been conjectured to be the culprit for phenomena such as voltage collapse in electric power systems[7] and species extinction in ecology.[8] It also plays a crucial role in governing the dynamics of shear flows in pipes and ducts at low Reynolds numbers.[9,10]

Given the importance of these phenomena, controlling transient chaos is a pressing issue. Some attempts to restore chaos in such scenarios have been made in the past. Yang *et al.*[5] maintained chaos in transiently chaotic regimes of one- and two-dimensional maps using small perturbations. Their method relied on accurate analytical knowledge of the dynamical system and required *a priori* phase-space knowledge of escape regions from chaos. Another method utilized the natural dynamics around the saddle,[11,12] where small regions near a chaotic saddle through which trajectories escape were identified. Then, a set of "target" points in these regions were found, which yield trajectories that can stay near the chaotic saddle for a relatively long time. When the solution trajectory falls in this escape region, it is perturbed to the nearest target point so that the trajectory can persist near the chaotic saddle for a long time. The identification of such escape regions and target points can be challenging and requires either an *a priori* computation of the probability distribution of escape times in different regions of the state space[11] or information from the return map constructed from local maxima or minima of a measured time series.[12] Such approaches become difficult for high-dimensional dynamical systems and have been illustrated for 2D maps/flows at the most. One particular control technique that worked for the 3D Lorenz system was described by Capeáns *et al.*[13] The method was based on finding a certain control-perturbation set in the phase space, called a "safe set," which avoids the escape of the trajectories to the fix-points. Identifying such a safe set can be prohibitively expensive computationally, and such safe sets may not exist for all dynamical systems. A useful alternative in such scenarios is to adopt control approaches that do not require *a priori* knowledge of the system's dynamics. Gadaleta and Dangelmayr[14] demonstrated an early application of such techniques, where they used reinforcement learning to stabilize unstable fixed points and periodic orbits in chaotic systems. We adopt a similar approach in this work, where a reinforcement learning-based autonomous controller continually perturbs the parameters of the Lorenz system to discover an optimal strategy for preventing transiently chaotic behavior of the system.

II. REINFORCEMENT LEARNING

In recent years, a machine-learning technique called deep Reinforcement Learning (RL) has shown great promise in control-optimization problems,[15] and it has been successfully used to uncover complex underlying physics in Navier–Stokes simulations of fish-swimming.[16] The aim of the present work is to illustrate the utility of deep RL in determining small control-perturbations to the parameters of the Lorenz system,[17] such that a sustained chaotic behavior is maintained despite the uncontrolled dynamics being transiently chaotic. In doing so, no prior analytical knowledge about the dynamical system and no special schemes to find escape regions, target points, and safe sets will be employed. The RL algorithm is able to autonomously determine an optimal strategy to restore chaos, by continually interacting with the dynamical system.

As depicted in Fig. 1, a reinforcement learning problem consists of five major elements—a learning agent, an environment described by a model $\boldsymbol{Y}$ (the Lorenz system in our case), state-space S, action-space A, and reward r_t. Initially, the RL agent interacts with its environment in a trial-and-error manner. At each time step t, the agent receives the current state s_t of the environment and selects an action a_t following a policy $\pi(a_t|s_t)$. This action allows the agent to perturb the state of the environment and move to a new state s_{t+1} by evaluating the given model $\boldsymbol{Y}$ of the environment. Upon affecting this transition, the agent is rewarded (or punished) with reward r_t. This process continues until the agent reaches a terminal state, at which point a new episode starts over. The return received from each episode is the discounted cumulative reward with the discount factor γ, which lies between 0 and 1. The discount factor makes it feasible to emphasize the importance of maximizing long-term rewards, which enables the agent to prefer actions that are beneficial in the long-term. The cumulative reward, $R(a_t|s_t)$, is given as

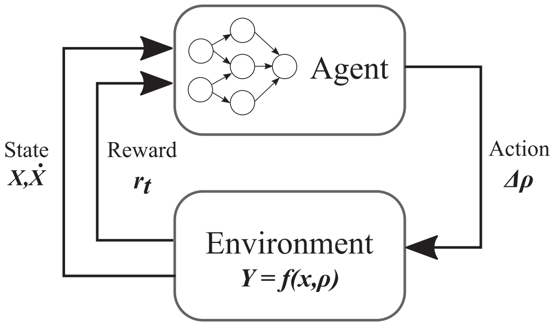

FIG. 1. Schematic illustrating the basic framework of a reinforcement learning problem. An agent continually perturbs the environment (which is the Lorenz system in our case) by taking an action and records the resulting states. The agent is rewarded when a desirable state is reached and punished otherwise.

$$R(a_t|s_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \tag{1}$$

III. TRAINING THE AGENT TO SUSTAIN CHAOS

The goal of the RL agent is to maximize the cumulative reward by discovering an optimal policy π^*. There are a variety of methods available for attaining this. We make use of Proximal Policy Optimization (PPO),[18] which is a type of policy Gradient Method (PGM).[19,20] A detailed description of the algorithm is provided in the supplementary material. PPO is suitable for continuous-control problems,[21] and it is simpler in its mathematical implementation compared to other PGM based RL algorithms.[22] Moreover, PPO requires comparatively little hyper-parameter tuning for use in a variety of different problems. The specific implementation of the algorithm that we used, PPO2, is available as part of the OpenAI stable-baselines library.[23] The ergodic and unsteady nature of chaotic dynamics necessitates the use of a version of PPO2 where the policy is defined by deep recurrent neural networks comprised of Long Short-Term Memory (LSTM) cells,[24] instead of traditional feed-forward neural networks. A brief description of LSTM cells and their advantages is provided in the supplementary material.

The environment for the Lorenz system is written in an OpenAI gym[25]-compatible python format and is provided as part of the supplementary material. The relevant equations are given as

$$\frac{dx}{dt} = \sigma(y - x), \tag{2a}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \tag{2b}$$

$$\frac{dz}{dt} = xy - \beta z. \tag{2c}$$

With $\sigma = 10$ and $\beta = 8/3$, $\rho = 28$ gives rise to chaotic trajectories, whereas transient chaos is found in the interval $\rho \in [13.93, 24.06]$.[26] Without any control implemented, the solution will converge to specific fix-points after a short transient, as shown in Fig. 2. The two fix-points in our case are given by $\mathrm{P}_+ = (7.12, 7.12, 19)$ and $\mathrm{P}_- = (-7.12, -7.12, 19)$.

We use reinforcement learning to prevent such a transient from chaotic- to fix-point solutions. This is done by perturbing the parameters in Eq. (2) $[\boldsymbol{\rho} = (\sigma, \rho, \beta)]$ by $\Delta\boldsymbol{\rho} = (\Delta\sigma, \Delta\rho, \Delta\beta)$, with $\Delta\boldsymbol{\rho} \in [-\boldsymbol{\rho}/10, \boldsymbol{\rho}/10]$. We clarify that these limits do not change with the "current" value of $\boldsymbol{\rho}$ but instead remain fixed at the initial value. The instantaneous value of the solution vector $\boldsymbol{X}(t) = (x, y, z)$ and its time-derivative (velocity) $\dot{\boldsymbol{X}}(t) = (V_x(t), V_y(t), V_z(t)) = (\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt})$ constitute the state space S for the RL algorithm. For training the RL agent to retain a chaotic trajectory, we utilize the fact that $|\boldsymbol{V}(t)|$ will decrease consistently as the solution converges to one of the fix-points, eventually becoming zero. On the other hand, $|\boldsymbol{V}(t)|$ will have a non-zero average value when the solution traces the chaotic attractor. Thus, whenever the agent determines suitable action values $\Delta\boldsymbol{\rho}$ for which $|\boldsymbol{V}(t)|$ is maintained above the predefined threshold value $V_0 = 40$, it is rewarded; otherwise, it is punished. In doing so over several iterations, the agent eventually learns to keep the trajectory chaotic. We remark that the method is robust against the choice of V_0 unless a very small value of V_0 is chosen. The strategy used here for selecting an appropriate V_0 is to use a value close to the ensemble average of velocity magnitude sampled from various instances of chaotic transients.

The reward allocated to the agent consists of two parts: a step-wise reward r_t provided at each time step and a one-time terminal reward $r_{terminal}$ given at the end of each episode. The two terms take the following form:

$$r_t = \begin{cases} 10, & V(t) > V_0, \\ -10, & V(t) \leqslant V_0, \end{cases} \tag{3a}$$

$$r_{terminal} = \begin{cases} -100, & \bar{r}_t < -2, \\ 0, & \bar{r}_t > -2. \end{cases} \tag{3b}$$

The average $\bar{r}_t$ is defined over the last 2000 time steps of an episode and facilitates learning to keep the trajectory chaotic over long periods of time. The training of the agent is divided into episodes of 4000 time steps each, with time step size $dt = 2e{-}2$. The RL agent is expected to learn suitable action values $\Delta\boldsymbol{\rho}$ for any state permissible by the system environment, such that the long-term reward accumulated is maximized.

The neural network architecture used for training the RL agent is shown in Fig. 3. The network consists of an input layer, an output layer, and three hidden layers in between. Figure 4 illustrates the training of the agent with time. The underlying neural network is trained for 2×10^5 time steps, which corresponds to 50 independent episodes in total, with each episode beginning with random values of the state variables $\boldsymbol{X}$ between -40 and 40; the corresponding values

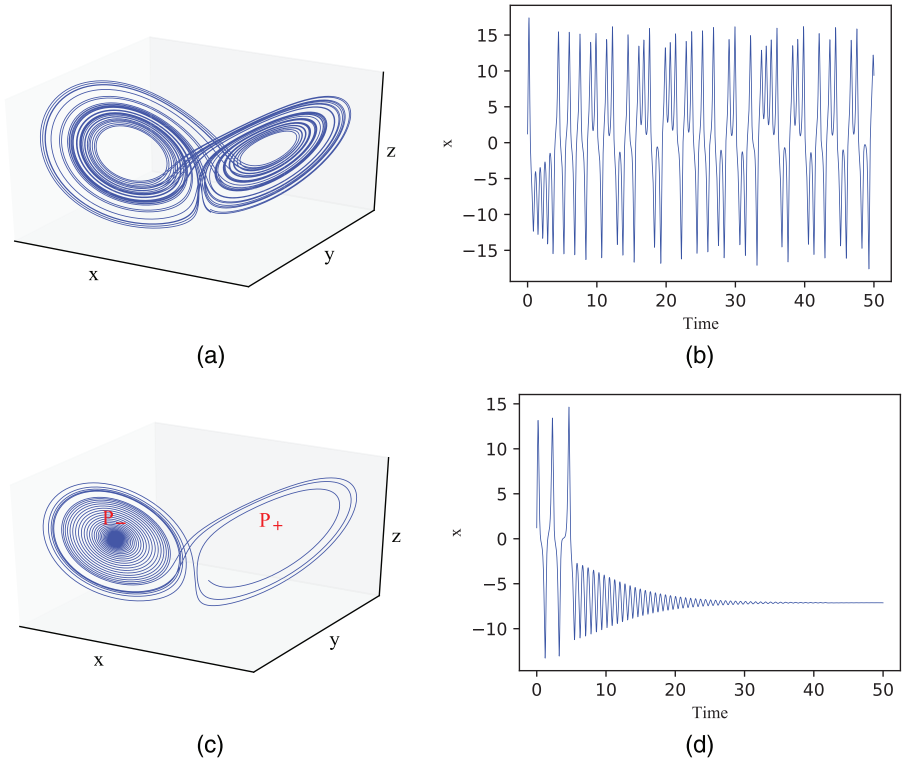

FIG. 2. Solution of the Lorenz system of equations in (a) and (b) the chaotic regime with $\rho = 28$ and (c) and (d) the transiently chaotic regime with $\rho = 20$. Panels (a) and (c) depict the solutions in the phase space, and the corresponding time-variation of the x coordinate is shown in panels (b) and (d). Note that the solution traverses a chaotic trajectory for $\rho = 28$, whereas it converges to P_- after a few chaotic transients for $\rho = 20$.

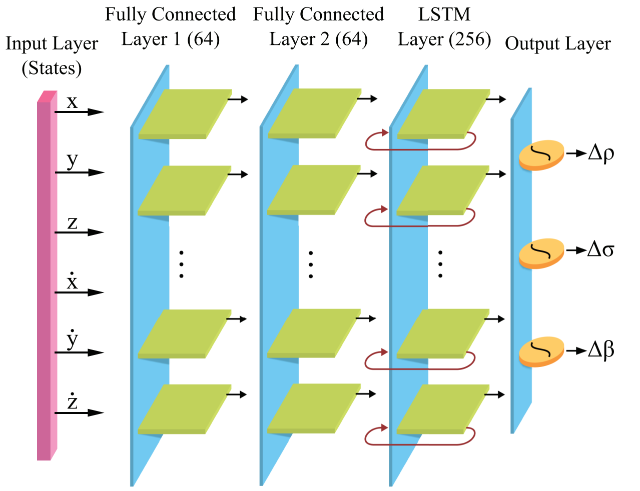

FIG. 3. A schematic diagram of the neural network architecture used in our study. The 6 state variables feed into a network with 2 fully connected layers consisting of 64 nodes each, followed by an LSTM layer comprising 256 cells. The output layer has three nodes corresponding to the three possible actions. Hyperbolic tangent (tanh) activation is used throughout the network.

for $\dot{\boldsymbol{X}}$ are determined using the Lorenz equations [Eq. (2)]. Initially, the solution keeps converging to the fix-points, since the network is unable to provide optimal action-decisions. After the network has trained for some time, it successfully learns the optimal actions for keeping the value of $|\boldsymbol{V}(t)|$ above V_0. As a consequence, the agent learns that the best way of maximizing reward is by maintaining the dynamics over the chaotic-attractor, which, although non-attracting for the given set of parameters, is a coexisting attractor of the system, along with the fix-points. In Fig. 4(b), we observe that the autonomous controller learns a distinct trend for ρ, which eventually suppresses the transiently chaotic behavior beyond $t = 2000$. However, the other two parameters σ and β do not exhibit a notable pattern. This indicates that the controller focuses primarily on the parameter that is best able to restore chaotic dynamics.

We note that the training procedure for the present study is not extremely demanding in terms of computational cost; a complete training run requires approximately 10 min on a regular laptop computer. The main difficulty arises when deciding the most appropriate form of the reward function, since variations in the formulation can lead to notably different outcomes.

IV. CONTROL STRATEGY DISCOVERED BY THE AUTONOMOUS AGENT

Figure 5 shows the distribution of the perturbation variable $\Delta\rho$ employed by the trained agent, which allows it to keep the dynamics on the chaotic-attractor. This distribution was obtained by plotting the controlled-trajectories for 400 random initial values for the variables x, y, and z, lying between -40 and 40. Note that a similar distribution was obtained for the other perturbation variables $\Delta\beta$ and $\Delta\sigma$. However, we find that an execution of the converged RL control-policy with $\Delta\beta$ and $\Delta\sigma$ explicitly set to zero does not make a difference in the control outcome; the agent is still able to maintain a chaotic trajectory. This may be attributed to the dominating magnitude of the parameter ρ compared to the other two parameters. The significance of ρ is also evident in Fig. 4(b) where a transition in the mean value of ρ from approximately 20 to 21 is observed when the RL agent becomes effective at keeping the dynamics chaotic. No such transition is evident for β and σ. We note that a mean value of 21 for ρ does not correspond to the stable chaotic regime of the Lorenz system, and solving the equations with constant $\rho = 21$ would still lead to transiently chaotic behavior. From the distribution shown in Fig. 5(a), we observe that the perturbation values for $\Delta\rho$ are predominantly negative in the region $\Re$, where $V_z = dz/dt < 0$, and positive elsewhere. We observe this correspondence by visualizing the velocity vectors in Fig. 5(b), which

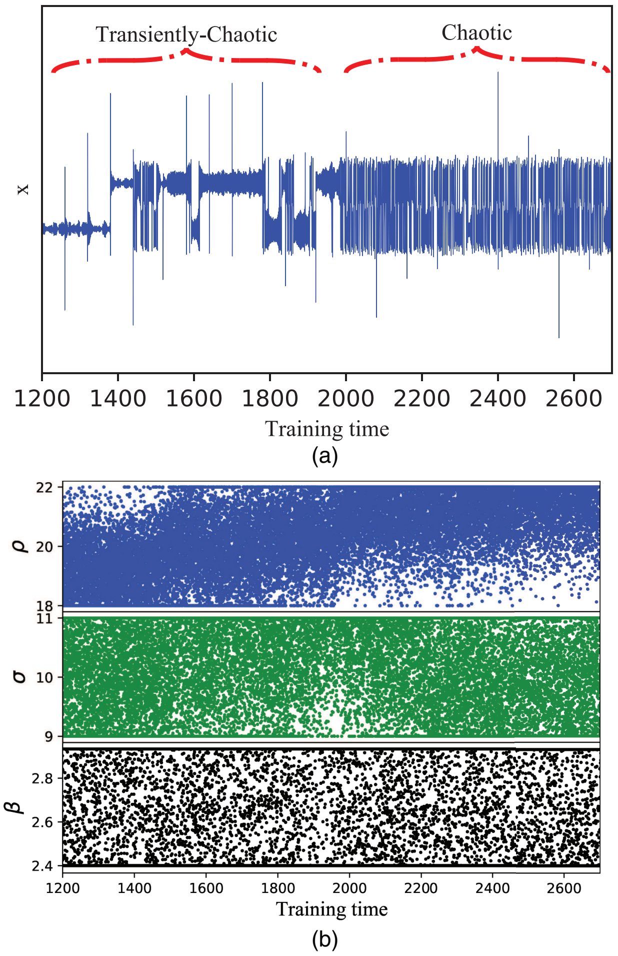

FIG. 4. (a) Training of the RL agent with time. Note that the trajectory is transiently chaotic until around $t = 2000$, beyond which the agent learns to take effective decisions to keep the solution trajectory chaotic for further instances. (b) Time-variation of the controlled parameters ρ, σ, and β. Note the gradual transition of the mean value of ρ, which corresponds with the eventual switch to chaotic behavior.

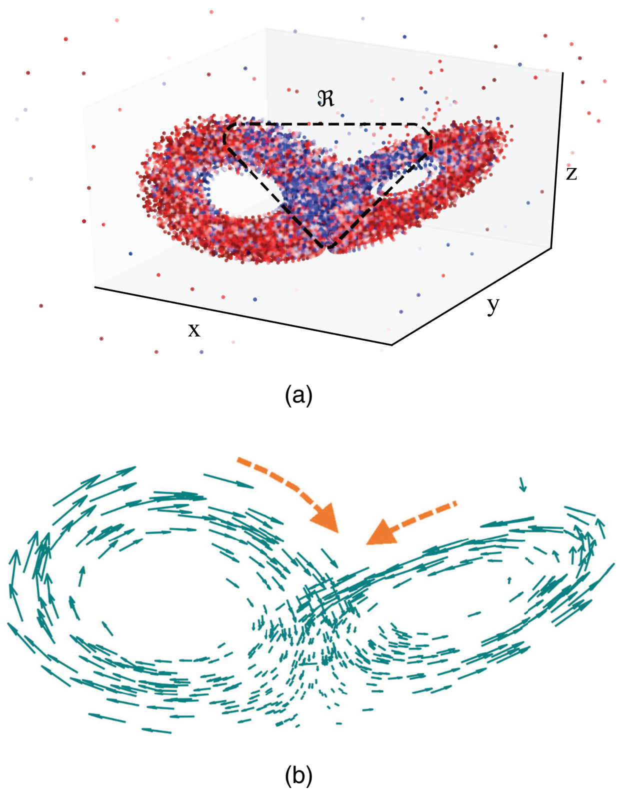

FIG. 5. (a) Distribution of the perturbation parameter $\Delta\rho$ learned by the RL agent to keep the dynamics on the chaotic-attractor. The red dots indicate locations where the perturbation values are positive, and the blue dots correspond to negative values. (b) Velocity vector ($\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}$) plot shown for the corresponding solution in the state space. Note that $\Delta\rho$ is predominantly negative in the region $\Re$ where $V_z = \frac{dz}{dt} < 0$.

indicate that the direction of motion within the region $\Re$ almost always results in a decrease of the z coordinate, i.e., $V_z < 0$. The overall effect of this control mechanism is to prevent the trajectory from spiraling into the fix-points.

Note that unlike other control-techniques, RL-based control requires no *a priori* analytical knowledge about the dynamical system regarding its escape regions, target points, and safe sets. The RL agent learns an optimal strategy π^* to prevent the transition from chaotic to fix-point solutions completely autonomously, by continually interacting with the environment defined by the Lorenz system of equations exhibiting transient chaos.

V. REVERSE ENGINEERING A CONTROL-LAW

Based on the strategy of the RL controller, we formulate a simple rule-based controller, which perturbs the parameter ρ by $-\rho/10$ whenever the trajectory visits the region $\Re$, i.e., whenever $V_z < 0$. All parameters remain unperturbed outside this region. The success of the rule-based binary-control is demonstrated in Fig. 6 (Multimedia view), where the uncontrolled trajectory (green)

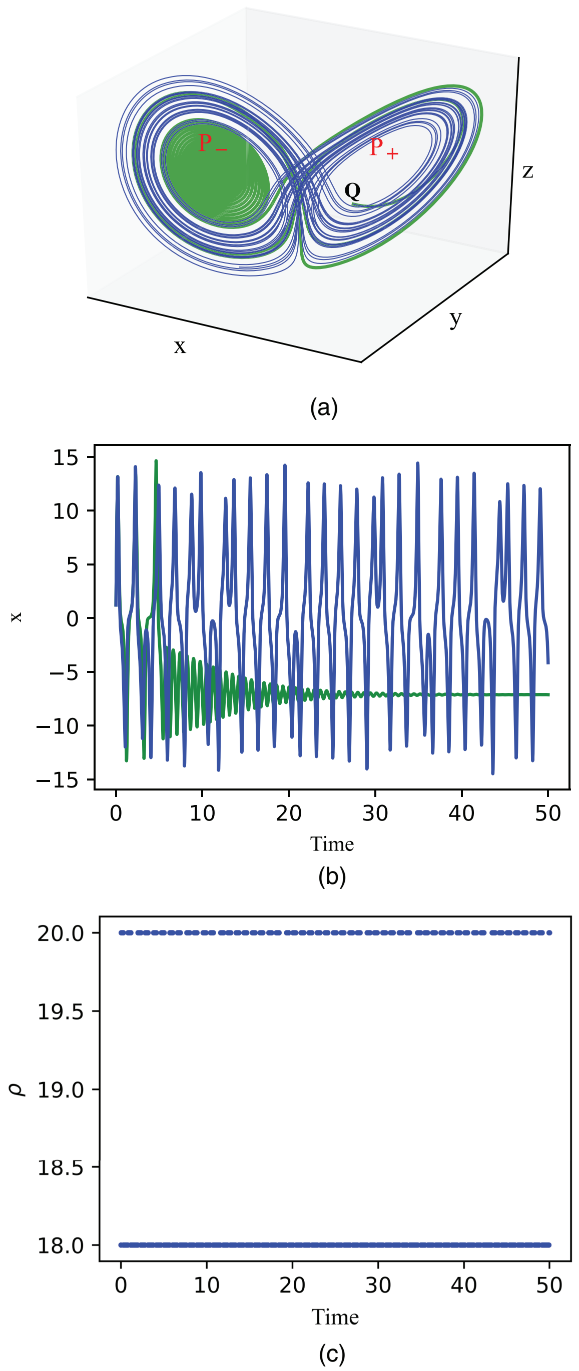

FIG. 6. (a) and (b) Comparison of the trajectory with and without the application of rule-based control. The blue trajectory corresponds to the controlled solution, starting from the initial condition $\mathbf{Q} = (1, 12, 9)$. The green uncontrolled solution starts from the same initial condition and spirals into the fix-point P_{-}. The corresponding animation is available in Supplementary Movie 1. (c) Time-variation of ρ for the rule-based control approach. Multimedia view: https://doi.org/10.1063/5.0002047.1

converges to the fix-point P$_-$, whereas its controlled counterpart (blue) remains chaotic. We remark that in the controlled scenario, ρ takes on two discrete values $\rho = 20.0$ and $\rho = 18.0$ [Fig. 6(c)], both of which lie within the range $\rho \in [13.93, 24.06]$ where transient chaos would be observed without active control. This demonstrates that autonomous strategies discovered by RL can be extremely useful for formulating simple control-laws in fairly complex dynamical systems.

VI. CONCLUSION

We have demonstrated the utility of deep reinforcement learning in restoring chaos for a transiently chaotic system. The learning algorithm autonomously discovers an effective strategy for perturbing the parameters of the Lorenz system to achieve its goal. We analyze the underlying strategy to formulate a simple control-law, which is able to sustain the chaotic trajectory even in the transiently chaotic parametric regime. Since transient chaos is a consequence of a catastrophic bifurcation (crisis),[27] our results pave the way for RL-enabled control of catastrophes in non-linear dynamical systems.

SUPPLEMENTARY MATERIAL

Please see the supplementary material for a discussion of the PPO algorithm and LSTM cells.

ACKNOWLEDGMENTS

This work was supported by the Department of Ocean and Mechanical Engineering at the Florida Atlantic University, as part of the start-up package of Siddhartha Verma.

REFERENCES

[1] I. T. Georgiou and I. B. Schwartz, "The slow invariant manifold of a conservative pendulum-oscillator system," Int. J. Bifurcation Chaos **6**, 673–692 (1996).

[2] H. Haken, "Chaos and order in nature," in *Chaos and Order in Nature* (Springer, 1981), pp. 2–11.

[3] J. C. Sackellares, L. D. Iasemidis, D.-S. Shiau, R. L. Gilmore, and S. N. Roper, "Epilepsy—When chaos fails," in *Chaos in Brain?* (World Scientific, 2000), pp. 112–133.

[4] A. L. Goldberger, D. R. Rigney, and B. J. West, "Chaos and fractals in human physiology," Sci. Am. **262**, 42–49 (1990).

[5] W. Yang, M. Ding, A. J. Mandell, and E. Ott, "Preserving chaos: Control strategies to preserve complex dynamics with potential relevance to biological disorders," Phys. Rev. E **51**, 102 (1995).

[6] C. Grebogi, E. Ott, and J. A. Yorke, "Crises, sudden changes in chaotic attractors, and transient chaos," Physica D **7**, 181–200 (1983).

[7] I. Dobson and H.-D. Chiang, "Towards a theory of voltage collapse in electric power systems," Syst. Control Lett. **13**, 253–262 (1989).

[8] K. McCann and P. Yodzis, "Nonlinear dynamics and population disappearances," Am. Nat. **144**, 873–879 (1994).

[9] K. Avila, D. Moxey, A. de Lozar, M. Avila, D. Barkley, and B. Hof, "The onset of turbulence in pipe flow," Science **333**, 192–196 (2011).

[10] T. Kreilos, B. Eckhardt, and T. M. Schneider, "Increasing lifetimes and the growing saddles of shear flow turbulence," Phys. Rev. Lett. **112**, 044503 (2014).

[11] I. B. Schwartz and I. Triandaf, "Sustaining chaos by using basin boundary saddles," Phys. Rev. Lett. **77**, 4740 (1996).

[12] M. Dhamala and Y.-C. Lai, "Controlling transient chaos in deterministic flows with applications to electrical power systems and ecology," Phys. Rev. E **59**, 1646 (1999).

[13] R. Capeáns, J. Sabuco, M. A. Sanjuán, and J. A. Yorke, "Partially controlling transient chaos in the Lorenz equations," Philos. Trans. R. Soc. Lond. A **375**, 20160211 (2017).

[14] S. Gadaleta and G. Dangelmayr, "Optimal chaos control through reinforcement learning," Chaos **9**, 775–788 (1999).

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," Nature **529**, 484 (2016).

[16] S. Verma, G. Novati, and P. Koumoutsakos, "Efficient collective swimming by harnessing vortices through deep reinforcement learning," Proc. Natl. Acad. Sci. U.S.A. **115**, 5849–5854 (2018).

[17] E. N. Lorenz, *The Essence of Chaos* (University of Washington Press, 1995).

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347 (2017).

[19] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99)* (MIT Press, Cambridge, MA, 1999), pp. 1057–1063.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018).

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," Nature **518**, 529 (2015).

[22] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," in *ICML'15: Proceedings of the 32nd International Conference on Machine Learning* (Microtome, 2015), Vol. 37, pp. 1889–1897.

[23] A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, see https://github.com/hill-a/stable-baselines for "Stable Baselines" (2018).

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).

[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," arXiv:1606.01540 (2016).

[26] J. L. Kaplan and J. A. Yorke, "Preturbulence: A regime observed in a fluid flow model of Lorenz," Commun. Math. Phys. **67**, 93–108 (1979).

[27] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, and C. Grebogi, "Predicting catastrophes in nonlinear dynamical systems by compressive sensing," Phys. Rev. Lett. **106**, 154101 (2011).