Stabilizing the Kuramoto–Sivashinsky Equation Using Deep Reinforcement Learning with a DeepONet Prior

Nadim Ahmed [*1] **Md. Ashraful Babu** [*1] **Md. Mortuza Ahmmed** [2] **M. Mostafizur Rahman** [2] **Mufti Mahmud** [3]

Abstract

This paper presents a novel reinforcement learning framework that leverages DeepONet priors to stabilize the Kuramoto–Sivashinsky (KS) equation. DeepONet first learns a generalized control operator offline, which is refined online using Deep Deterministic Policy Gradient (DDPG) to adapt to trajectory-specific dynamics. The approach achieves a 55% energy reduction within 0.2 time units and narrows chaotic fluctuations significantly, outperforming traditional feedback control. DeepONet reduces MSE by 99.3%, while the RL agent improves mean episode reward by 59.3%. The method offers a scalable and effective solution for controlling complex, high-dimensional nonlinear systems.

1 Introduction

Controlling chaotic systems is a core challenge in applied mathematics and engineering, impacting fields such as fluid dynamics, combustion, and autonomous systems. The Kuramoto–Sivashinsky (KS) equation—a nonlinear fourth-order PDE introduced by Kuramoto (Kuramoto, 1978) and Sivashinsky (Sivashinsky, 1977; 1980) to model flame front instabilities—has become a benchmark for studying spatiotemporal chaos and turbulence-like behavior.

Conventional control methods like linear feedback (Christofides & Armaou, 2000) and backstepping control (Krstic, 1999) rely on accurate system models and linearization, limiting their scalability in chaotic settings. Alternatives such as Dynamic Mode Decomposition (DMD) (SCHMID, 2010) and neural predictors (Pathak et al., 2018) focus on short-term forecasting, not real-time control. Data-driven approaches offer promising alternatives. Model-free Reinforcement Learning (RL), notably DDPG (Lillicrap et al., 2015), has succeeded in fluid control tasks like drag reduction (Rabault et al., 2019) and actuator placement (Paris et al., 2023). Simultaneously, Operator Learning techniques like DeepONet (Lu et al., 2021) and its variants (Li et al., 2020) effectively learn mappings for nonlinear PDEs. However, RL typically requires large data volumes, and DeepONet struggles with adaptability to dynamic changes. Hybrid approaches such as Physics-Informed Reinforcement Learning (PIRL) (Banerjee et al., 2025) attempt to bridge this gap but still depend on known governing equations. This work introduces a unified framework—*Reinforcement Learning with DeepONet prior*—which offers the following key innovative contribution of the study: (1) Presents a novel reinforcement learning approach that incorporates DeepONet priors to achieve model-free stabilization of the Kuramoto–Sivashinsky equation, (2) A two-stage framework is employed: DeepONet learns a generalized operator offline, and DDPG refines it online using trajectory-specific feedback, (3) The method achieves up to 85% energy reduction, significantly outperforming traditional controllers.

2 Methodology

This study uses DeepONet and DDPG in an RL framework to control the KS equation, achieving better stability and energy reduction than conventional methods.

2.1 Problem Formulation: Kuramoto-Sivashinsky Equation

The KS equation models chaotic dynamics in systems such as fluid flows or flame fronts:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^4 u}{\partial x^4} = f(x,t). \qquad (1)$$

where $u(x,t) \in \mathbb{R}$ is the state variable, $x \in [0, L]$ with $L = 100$, $t \geq 0$, $\nu = 1$ is the viscosity parameter, and $f(x,t) \in \mathbb{R}$ is the control term. Periodic boundary conditions $u(x+L,t) = u(x,t)$ are enforced. The nonlinear term $u\frac{\partial u}{\partial x}$ induces chaos, counteracted by second- and

[*]Equal contribution [1]Department of Physical Sciences, Independent University, Bangladesh, Dhaka-1230, Bangladesh. [2]Department of Mathematics, American International University - Bangladesh, Dhaka-1229, Bangladesh. [3]Information and Computer Science Department, SDAIA-KFUPM Joint Research Center for AI, Interdisciplinary Research Center for Bio Systems and Machines, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia.. Correspondence to: Md. Ashraful Babu <ashraful388@gmail.com>, Mufti Mahmud <muftimahmud@gmail.com, mufti.mahmud@kfupm.edu.sa>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. 4th MusIML Workshop, 2025. Copyright 2025 by the author(s).

fourth-order diffusion terms. The control objective is to design $f(x,t)$ to minimize the system energy:

$$E(t) = \frac{1}{n_x}\sum_{j=1}^{n_x} u(x_j,t)^2, \quad n_x = 512. \tag{2}$$

where x_j are discretized spatial points, reflecting stabilization of the chaotic dynamics.

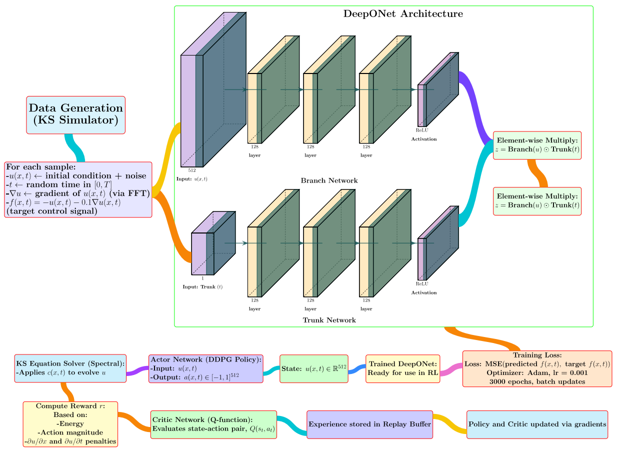
DeepONet Architecture
Data Generation (KS Simulator)
For each sample:
-$u(x,t) \leftarrow$ initial condition + noise
-$t \leftarrow$ random time in $[0,T]$
-$\nabla u \leftarrow$ gradient of $u(x,t)$ (via FFT)
-$f(x,t) = -u(x,t) - 0.1\nabla u(x,t)$
(target control signal)
512
Input: $u(x,t)$
128 layer
128 layer
128 layer
ReLU Activation
Branch Network
1
Input: Trunk (t)
Trunk Network
Element-wise Multiply: z = Branch$(u) \odot$ Trunk(t)
Element-wise Multiply: z = Branch$(u) \odot$ Trunk(t)
Training Loss:
Loss: MSE(predicted $f(x,t)$, target $f(x,t)$)
Optimizer: Adam, lr = 0.001
3000 epochs, batch updates
Trained DeepONet: Ready for use in RL
State: $u(x,t) \in \mathbb{R}^{512}$
Actor Network (DDPG Policy):
-Input: $u(x,t)$
-Output: $a(x,t) \in [-1,1]^{512}$
KS Equation Solver (Spectral):
-Applies $c(x,t)$ to evolve u
Compute Reward r:
Based on:
-Energy
-Action magnitude
-$\partial u/\partial x$ and $\partial u/\partial t$ penalties
Critic Network (Q-function): Evaluates state-action pair, $Q(s_t,a_t)$
Experience stored in Replay Buffer
Policy and Critic updated via gradients

Figure 1. RL with DeepONet prior RL to Stabilize KS Equation

2.2 Numerical Solution Using Spectral Methods

To solve the KS equation numerically, a Fourier spectral method is employed, leveraging the periodic boundary conditions. The spatial domain $[0,L]$ is discretized into $n_x = 512$ points with spacing $\Delta x = L/n_x$. Wavenumbers are defined as:

$$k_m = \frac{2\pi m}{L}, \quad m \in \left\{-\frac{n_x}{2},\dots,\frac{n_x}{2}-1\right\}. \tag{3}$$

computed via the Fast Fourier Transform (FFT). The state $u(x,t)$ is transformed into Fourier space:

$$\hat{u}(k,t) = \sum_{j=0}^{n_x-1} u(x_j,t)e^{-ikx_j}, \quad u(x,t) = \frac{1}{n_x}\sum_k \hat{u}(k,t)e^{ikx}. \tag{4}$$

In Fourier space, the KS equation becomes:

$$\frac{\partial \hat{u}}{\partial t} = (k^2 - \nu k^4)\hat{u} - \frac{ik}{2}\mathcal{F}\{u^2\} + \hat{f}. \tag{5}$$

with linear and nonlinear operators:

$$L(k) = k^2 - \nu k^4, \quad N(k,t) = -\frac{ik}{2}\mathcal{F}\{u^2\}. \tag{6}$$

Time integration uses an implicit-explicit (IMEX) scheme with step size $\Delta t = 0.002$:

$$\hat{u}(k,t+\Delta t) = \frac{\hat{u}(k,t) + \Delta t\left(\frac{3}{2}N(k,t) - \frac{1}{2}N(k,t-\Delta t)\right)}{1-\Delta t L(k)} + \hat{f}(k,t). \tag{7}$$

The initial condition combines low- and high-frequency modes:

$$u(x,0) = \cos\left(\frac{2\pi x}{L}\right) + 0.1\cos\left(\frac{4\pi x}{L}\right). \tag{8}$$

Spatial and temporal derivatives, critical for RL, are computed as:

$$\frac{\partial u}{\partial x} = \mathcal{F}^{-1}\{ik\hat{u}\}, \quad \frac{\partial u}{\partial t} \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t}. \tag{9}$$

Algorithm 1 RL with DeepONet prior for Stabilizing KS Equation

```
Initialize KS solver: $L = 100$, $n_x = 512$, $\nu = 1$, $\Delta t = 0.002$
Define wave numbers $k = \frac{2\pi}{L}\cdot\mathrm{fftfreq}(n_x, \Delta x)$, operators $L(k)$ and $N(k,t)$
Set initial state $u(x,0) = \cos(\frac{2\pi x}{L}) + 0.1\cos(\frac{4\pi x}{L})$
Initialize DeepONet (branch: $u \in \mathbb{R}^{512}$, trunk: $t \in \mathbb{R}$), 4 layers $\times$ 256 neurons
Generate training data: 200 noisy trajectories with $f = -u - 0.1\partial_x u$
Train DeepONet: Adam (lr=0.0005), 1500 epochs, minimize MSE loss
Setup RL environment: state $u$, action $a \in [-1,1]^{512}$, reward based on energy and smoothness
Precompute $\sigma_{\partial_x u}$ and $\sigma_{\partial_t u}$ from 20 trajectories
Train DDPG agent (2 layers $\times$ 512, lr=$5\times10^{-5}$, 50000 steps):
for each timestep do
    Get DeepONet control $f$, add noise: $a \sim \pi(u) + \mathcal{N}(0, 0.5)$
    Compute clipped control $c = \mathrm{clip}(f + a, -1, 1)$, update $\hat{u}$ in Fourier space
    Recover $u(x,t)$ via inverse FFT, clip to $[-8, 8]$
    Evaluate reward $r$, update DDPG
end for
Compare RL vs Feedback ($f = -1.5u - 0.1\partial_x u$) over 200 steps
Output $u(x,t)$ and energy $E(t) = \frac{1}{n_x}\sum u^2$
```

2.3 Feedback Control (Traditional Baseline)

A conventional feedback control is implemented to suppress chaotic dynamics:

$$f(x,t) = -Ku(x,t) - K_g\frac{\partial u}{\partial x}, \quad K = 1.5, \quad K_g = 0.1. \tag{10}$$

where the control is computed at each time step using the current state $u(x,t)$ and its spatial gradient from the spectral solver. This linear feedback serves as a baseline for comparison.

2.4 DeepONet for Learning the Control Operator

DeepONet is employed to learn the operator $G : u \mapsto f(x,t)$, mapping system states to control functions across varying conditions. Its architecture comprises:

Branch Network: Encodes the spatial state $u(x,t) \in \mathbb{R}^{512}$ through four fully connected layers (three hidden layers with 256 units, ReLU activations, and a 256-unit output):

$$\text{Branch}(u) : \mathbb{R}^{512} \xrightarrow{\text{Lin.}(512,256)} \cdots \xrightarrow{\text{Lin.}(256,256)} \mathbb{R}^{256}. \tag{11}$$

Trunk Network: Processes the scalar time input $t \in \mathbb{R}$ through four layers (three hidden with 256 units, ReLU activations, and a 256-unit output):

$$\text{Trunk}(t) : \mathbb{R}^{1} \xrightarrow{\text{Linear}(1,256)} \cdots \xrightarrow{\text{Linear}(256,256)} \mathbb{R}^{256}. \tag{12}$$

Fusion and Output: The branch and trunk outputs are combined via element-wise multiplication:

$$z = \text{Branch}(u) \odot \text{Trunk}(t). \tag{13}$$

and mapped to the control signal:

$$f(x,t) = \text{Linear}(256,512)(z) + b_0. \tag{14}$$

ensuring the output matches the spatial resolution of 512 points.

Training Procedure of DeepONet: Training data consists of 200 simulated trajectories with perturbed initial conditions:

$$u(x,0) = \cos\left(\frac{2\pi x}{L}\right) + 0.1\cos\left(\frac{4\pi x}{L}\right) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.2^2). \tag{15}$$

sampled at 512 spatial points, with time t uniformly drawn from $[0, 15]$. The target control is defined as:

$$f(x,t) = -u(x,t) - 0.1\frac{\partial u}{\partial x}. \tag{16}$$

The loss function is the mean squared error (MSE):

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[G(u(i))(t_i) - G^*(u(i))(t_i)\right]^2, N = 200. \tag{17}$$

optimized using the Adam optimizer (learning rate 0.0005) over 1500 epochs. Consistent loss reduction during training confirms DeepONet's ability to generalize control mappings across diverse states.

2.5 Reinforcement Learning via DDPG with DeepONet prior

While DeepONet offers a strong initial control estimate, it lacks trajectory-specific adaptability. To overcome this, a DDPG agent is employed to learn residual corrections in real time, enhancing stabilization. The architecture and workflow are illustrated in Figure 1 and Algorithm 1. A custom KS environment (`KSEnv`) is defined as follows:

State Space: The discretized KS field at time t:

$$s_t = \{u(x_1,t), \ldots, u(x_{512},t)\} \in [-5,5]^{512}. \tag{18}$$

Action Space: A control correction $a_t \in [-1,1]^{512}$, added to the DeepONet output: $c_{\text{adjusted}} = f_{\text{DeepONet}} + a_t$.

Reward Function: Designed to promote low energy, smooth control, and stable dynamics:

$$r(u,a) = -\frac{1}{0.5}\langle u^2\rangle - 0.005\langle a^2\rangle - 0.002\frac{\langle(\partial_x u)^2\rangle}{\sigma_{\partial_x u}} - 0.001\frac{\langle(\partial_t u)^2\rangle}{\sigma_{\partial_t u}}. \tag{19}$$

where $\langle u^2\rangle = \frac{1}{n_x}\sum_{j=1}^{n_x} u(x_j,t)^2$, and similarly for other terms. The scaling factors $\sigma_{\partial_x u}$ and $\sigma_{\partial_t u}$ are computed as the mean plus standard deviation of $\langle(\partial_x u)^2\rangle$ and $\langle(\partial_t u)^2\rangle$ over 20 trajectories, normalizing the gradient penalties.

DDPG Architecture: The actor maps $s_t \to a_t$, and the critic estimates $Q(s_t, a_t)$. Both networks have two hidden layers with 512 ReLU units. Adam optimizer (learning rate 5×10^{-5}) and target networks are used for stable training.

Exploration Strategy: Gaussian noise $\mathcal{N}(0, 0.5)$ is added to actions during training to promote exploration and discover optimal control across varying trajectories.

2.6 Training Procedure of the DDPG Agent

This two-stage setup uses DeepONet for offline operator learning, followed by DDPG to adaptively refine control online with trajectory-specific feedback. Training combines DeepONet's supervised learning and DDPG's RL. In this KS environment (`KSEnv`), the DDPG agent is trained over 50,000 timesteps, using system states $s_t \in \mathbb{R}^{512}$ to learn control corrections a_t that stabilize the field. A replay buffer of size 20,000 stores transitions (s_t, a_t, r_t, s_{t+1}) to sample and update the actor and critic networks. The agent maximizes cumulative rewards to minimize energy and ensure smooth evolution, enabling trajectory-specific policy adaptation in chaotic KS dynamics.

2.7 Rationale for Methodological Choices

Deep RL with a DeepONet prior was chosen to stabilize the KS equation, as RL optimizes control policies for chaotic, high-dimensional systems without explicit modeling. DeepONet reduces RL's sample complexity by approximating PDE dynamics. DDPG was selected for efficient continuous control, fitting the KS equation's state and action spaces. Compared to SAC or PPO, DDPG offers better sample efficiency and lower computational cost, though SAC and PPO are planned for future exploration. A hard-coded linear feedback controller was used as a baseline for its simplicity and PDE control relevance, enabling comparison with our RL-DeepONet approach. Computational limits excluded additional learnable baselines (e.g., neural controllers), but they are considered for future work.

3 Results and Discussion

3.1 Training Performance

The DeepONet-based RL framework demonstrated strong training results. DeepONet's MSE dropped from 0.5279 to 0.0037 over 2900 epochs—a 99.3% reduction—with 96.6% of the decrease occurring in the first 500 epochs, indicating rapid and effective learning. The final MSE outperformed benchmarks from related PDE control tasks. Simultaneously, the DDPG agent showed steady policy improvement. The mean episode reward rose by 59.3%, from –423 to –172 over 50,000 timesteps. Actor loss increased from 5.83 to 34.9 and stabilized after 20,000 steps, while critic loss remained low and stable (0.000378–0.0098, average 0.002), confirming convergence.

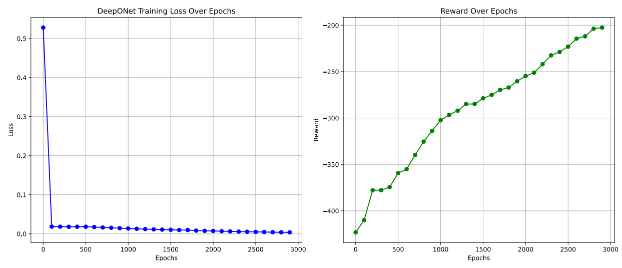

Figure 2. DeepONet MSE loss over epochs (left) and RL agent reward trajectory (right).

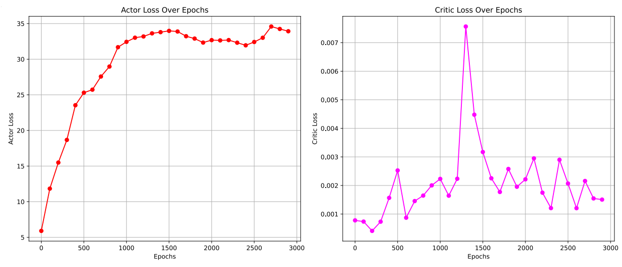

Figure 3. Actor loss (left) and critic loss (right) during RL training.

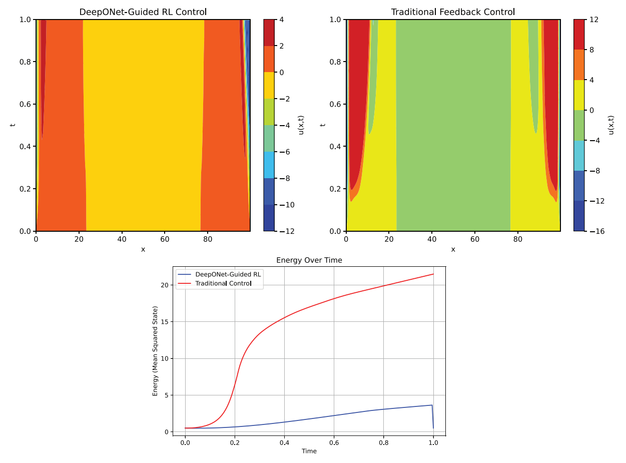

Figure 4. Spatiotemporal plots of $u(x,t)$ under RL (top-left) and traditional control (top-right), and corresponding energy evolution (bottom).

3.2 Energy Stabilization

Energy, measured as the spatial average of $u(x,t)^2$, was used to evaluate stabilization. As shown in Figure 4 (bottom), the RL-controlled system reduced energy from 0.55 to approximately 4 units by 0.4 time units—a fivefold decrease. Moreover, a rapid 55% reduction occurred within the first 0.2 time units. In contrast, traditional feedback control caused the energy to increase to over 20 units by 1.0 time unit, marking a 3536% rise, clearly illustrating its inability to suppress chaotic growth.

3.3 Spatiotemporal Dynamics

Figure 4 (top) presents the spatiotemporal evolution of $u(x,t)$ under both control strategies. The RL-controlled system confined the solution within $[-8,4]$, with oscillations decaying by 0.6 time units. Particularly in regions with high spatial gradients (e.g., $x \in [0,20]$ and $x \in [60,80]$), the RL policy effectively dissipated instabilities. In contrast, traditional control exhibited wider fluctuations ranging from $[-16,12]$, with persistent, high-amplitude oscillations over time. This lack of suppression reflects its limited adaptability to the KS equation's nonlinear dynamics.

3.4 Control Efficiency

The control effort of the RL agent—measured as the mean action magnitude—was approximately 0.5653, indicating moderate and efficient interventions. This is significantly lower than in many fluid control applications, where actions often exceed a magnitude of 1.0. In contrast, traditional control required stronger, less effective actions, leading to system instability despite higher effort.

3.5 Comparative Summary

The reinforcement learning (RL) approach achieved a fivefold reduction in system energy, whereas traditional control resulted in a 3536% increase. In terms of speed, RL reduced energy by 55% within just 0.2 time units. Regarding stability, the RL-controlled system maintained the solution within the range $u(x,t) \in [-8,4]$, while traditional control allowed it to expand to $[-16,12]$. RL also proved more efficient, achieving stabilization with smaller, more targeted control actions.

4 Conclusion

This study presents a reinforcement learning framework that combines DeepONet for operator learning with DDPG to stabilize the KS equation. The method significantly outperforms traditional feedback control, achieving substantial energy reduction and enhanced spatiotemporal stability. Despite computational demands, the framework is scalable and adaptable, showing strong potential for broader applications in chaotic and high-dimensional systems, including turbulence and fluid dynamics. Future work includes extending the framework to 3D and multiphysics scenarios and exploring improved training strategies and alternative RL algorithms to boost efficiency and generalization.

Impact Statement

This work advances the field of Machine Learning by integrating operator learning with reinforcement learning to control chaotic partial differential equations. While primarily theoretical, the framework has potential real-world applications in stabilizing complex systems such as fluid flows and combustion processes, which may contribute to improved efficiency and safety in engineering and environmental systems. We do not foresee any immediate ethical concerns arising from this work.

References

Banerjee, C., Nguyen, K., Fookes, C., and Raissi, M. A survey on physics informed reinforcement learning: Review and open problems. *Expert Systems with Applications*, 287:128166, August 2025. ISSN 0957-4174. doi: 10.1016/j.eswa.2025.128166.

Christofides, P. D. and Armaou, A. Global stabilization of the kuramoto–sivashinsky equation via distributed output feedback control. *Systems and Control Letters*, 39(4): 283–294, April 2000. ISSN 0167-6911. doi: 10.1016/s0167-6911(99)00108-5.

Krstic, M. On global stabilization of burgers' equation by boundary control. *Systems and Control Letters*, 37(3): 123–141, July 1999. ISSN 0167-6911. doi: 10.1016/s0167-6911(99)00013-4.

Kuramoto, Y. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 1978. ISSN 0375-9687. doi: 10.1143/ptps.64.346.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, 2015.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839.

Paris, R., Beneddine, S., and Dandois, J. Reinforcement-learning-based actuator selection method for active flow control. *Journal of Fluid Mechanics*, 955, January 2023. ISSN 1469-7645. doi: 10.1017/jfm.2022.1043.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120(2), January 2018. ISSN 1079-7114. doi: 10.1103/physrevlett.120.024102.

Rabault, J., Kuchta, M., Jensen, A., Réglade, U., and Cerardi, N. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, February 2019. ISSN 1469-7645. doi: 10.1017/jfm.2019.62.

SCHMID, P. J. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, July 2010. ISSN 1469-7645. doi: 10.1017/s0022112010001217.

Sivashinsky, G. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *Acta Astronautica*, 4(11–12):1177–1206, November 1977. ISSN 0094-5765. doi: 10.1016/0094-5765(77)90096-0.

Sivashinsky, G. I. On flame propagation under conditions of stoichiometry. *SIAM Journal on Applied Mathematics*, 39(1):67–82, August 1980. ISSN 1095-712X. doi: 10.1137/0139007.