List 4: Reinforcement Learning. Deep learning control of physics-informed neural networks for fast actuation in uncertain environments

Jostein Barry-Straume
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
`jostein@vt.edu`

1 Introduction

Why is this problem important to solve? Scientific Machine Learning (SciML) has arisen as a replacement to traditional numerical discretization methods. The main driving force behind this replacement is neural networks (NNs), largely due to their success in natural language processing (NLP) and computer vision (CV) problems [1]. As a vehicle to approximate the solution to a given partial differential equation (PDE) or ordinary differential equation (ODE), NNs offer a mesh-free approach via auto differentiation, and break the curse of dimensionality [2]. Combining scientific computing and ML, SciML offers the potential to improve "predictions beyond state-of-the-art physical models with smaller number of samples and generalizability in out-of-sample scenarios" [3].

Physics-informed neural networks (PINNs) are networks that solve supervised learning tasks while respecting the properties of physical laws [4]. Since their introduction, PINNs have been leveraged to solve a wide swath of problems including, but not limited to, inverse problems [2, 5–9], fractional differential equations [10], and stochastic differential equations [11–14]. Some recent work has began exploring the application of PINNs with regards to optimal control problems [15–18]. However, these prior approaches tackle control problems by merely feeding control data to a PINN.

Describe an important research problem. A new framework of PINNs presented in this paper, hereafter known as Control PINNs, offers a novel approach to solving open loop and closed loop optimal control problems. Control PINNs simultaneously solve the learning tasks of both the system state, and its respective optimal control, without the need for a priori controller data nor an external controller. Moreover, Control PINNs can find optimal control solutions to complex computational scientific problems more efficiently. Futhermore, Control PINNs can be utilized as agents in deep reinforcement learning (DRL) [19]. In DRL, finding the state of a robot after a given action may require solving a number of physical equations (e.g. equation of motion and balance of force). This issue can be circumvented by leveraging PINNs as an agent because PINNs penalize any deviation from given physical constraints.

We begin in section 2 by providing a literature review of background information and prior approaches. Section 3 covers the methodology of the novel approach that this paper offers as contribution. Section 4 validates the methodology via implementation of an analytical toy problem. Section 5 details a road map for future work.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

2 Background

2.1 Optimal Control

Hairer and Wanner provide an overview of control problems, and more specifically, optimal control problems in [20]. It is common for optimal control theory problems to be presented as ordinary differential equations of the form $y' = f(y, u)$ where $u(x)$ represents a set of controls and is applied such that a given constraint, say $0 = g(y, u)$, is satisfied by the solution while simultaneously minimizing a given cost function [20].

2.2 Physics-Informed Neural Networks

Cuomo *et* al. provide a comprehensive overview of Physics-Informed Neural Networks (PINNS) in [19]. This survey paper sheds light on how most literature involving PINNs deals with customizing PINNs "through different activation functions, gradient optimization techniques, neural network structures, and loss function structures" [19]. *The following excerpt from [19] highlights the important link between PINNs and reinforcement learning:*

> "PINNs can be used as a tool for engaging deep reinforcement learning (DRL) that combines reinforcement learning (RL) and deep learning. RL enables agents to conduct experiments to comprehend their environment better, allowing them to acquire high-level causal links and reasoning about causes and effects [21]. The main principle of reinforcement learning is to have an agent learn from its surroundings through exploration and by defining a reward [22]. In the DRL framework, the PINNs can be used as agents. In this scenario, information from the environment could be directly embedded in the agent using knowledge from actuators, sensors, and the prior-physical law, like in a transfer learning paradigm" [19].

Raissi et al. introduce the novel physics informed neural networks (PINNs) in [4]. Raissi et al. define PINNs to be networks that solve supervised learning tasks while respecting the properties of physical laws. In [4], PINNs are utilized to solve data-driven problems, and data-driven discovery of partial differential equations.

In their introductory paper of [4], PINNs function by minimizing a mean-squared-error loss function involving both data from the boundary conditions , and the enforced physical equations of the given problem. The general algorithm for a Physics-Informed Neural Network (PINN) is as follows below in algorithm 1.

Algorithm 1: The PINN algorithm for solving differential equations [2]

Result: Minimize loss function $L(\theta; T)$

1. Construct neural network $\hat{u}(x; \theta)$ with parameters θ.
2. Specify two training sets T_f and T_b for the equation and boundary/initial conditions.
3. Specify a loss function by summing the weighted L^2 norm of both the PDE equation and boundary condition residuals.
4. Train the neural network to find the best parameters θ^* by minimizing the loss function $L(\theta; T)$.

Willard *et* al. provide a structured overview of physics-based modeling approaches with machine learning (ML) techniques [3]. Moreover, Willard *et* al. summarize current areas of application with regard to science-guided ML [3]. Willard *et* al. describe current methodologies of constructing physics-guided ML models and hybrid physics-ML frameworks [3]. Consequently, Willard et al. have compiled a taxonomy of existing techniques, and as such have shed light on knowledge gaps and provided a foundation for new ideas to spring forth.

According to Willard *et* al., the five classes of methodologies to merge principles of physics-based modeling with ML are: 1.) Physics-guided loss function 2.) Physics-guided initialization 3.) Physics-guided design of architecture 4.) Residual modeling 5.) Hybrid physics-ML models.

In [23], Nellikkath and Chatzivasileiadis show that by “combining the [Karush-Kuhn-Tucker] conditions with the neural network, the physics-informed neural network achieves higher accuracy while utilizing substantially fewer data points." Moreover, the duo expanded on their previous work in [23] regarding “worst-case guarantees to cover the physics-informed neural networks (PINNs), and [...] show that PINNs result in lower worst-case violations than conventional neural networks."

2.3 Optimal Control with Physics-Informed Neural Networks

This section describes prior approaches to solve the proposed problem. In [15], Hwang *et* al. propose a two stage framework for solving PDE-constrained control problems using operator learning. They first train an autoencoder model, and then infer the optimal control by fixing the learnable parameter and minimizing their objective function. One strength of their approach is the ability to apply their framework to both data-driven and data-free cases. The main downside to their approach is the two stage nature of the framework, as the control is found only after a surrogate model has been trained. Success is measured through tracking the "values of relative errors on test data in each experiment," in addition to visual inspection of the trained solution operators [15].

In [16], Chen *et* al. train an input convex recurrent neural network. Subsequently, they then solve a convex model predictive control (MPC) problem. The main strength of the approach in [16] is the guarantee of an optimal solution, thanks to the convex nature of the trained model. The main limitation is similar to [15, 16], in that they employ a two stage framework of system identification and controller design [16]. Success is evaluated by the "performance of both algorithms on three randomly selected fixed random seeds for four tasks" [16]. The average performance, with corresponding standard deviation, is then visually plotted for visual confirmation of success.

In [17], Antonelo *et* al. introduce a new framework called Physics-Informed Neural Nets for Control (PINC). PINC uses data from the control action u, and initial state $y(0)$, to solve an optimal control problem. One strength of this approach is the ability to "run for an indefinite time horizon [...] without significant deterioration of network prediction" [17]. A significant limitation of this approach is offline learning the control separately from the solution operator. In other words, PINC is essentially a PINN that is amenable to being trained on data of the controller data, instead of learning the optimal control itself. Success is evaluated through Mean Squared Error (MSE) validation error for the Van der Pol Oscillator problem.

In [18], Mowlavi and Nabi conduct an evaluation of the comparative performance between traditional PINNs and classic direct-adjoint-looping (DAL) to solve optimal control problems. Similar to Antonelo *et* al. [17], Mowlavi and Nabi separate the optimal control problem into two sub-problems. At each state of the system, the PDE is solved with one neural network. That information is then used by another neural network to solve for the optimal control at that given state of the PDE. Afterwards, the adjoint PDE (lambda term) is solved in backwards time [18].

The strength of Mowlavi's and Nabi's approach is achieving one of their main goals of comparing "the pros and cons of the PINN and DAL frameworks for solving PDE-constrained optimal control problems, so that the novel PINN approach can be placed in the context of the mature field of PDE-constrained optimization" [18]. Success is measured via validation and evaluation steps. Validation is done by monitoring residual, boundary, and initial loss components with a known a priori solution. Evaluation is done by comparing the control cost objective with an a priori solution found by a high-fidelity numerical solver.

One limitations of this approach is using the more easily solvable steady state Navier Stokes, instead of unsteady state Navier Stokes. Moreover, manual derivation is used in their DAL approach, which is unneccesary because DAL can use automatic differentiation (AD). Consequently, to a certain extent, [18] is solving the optimal control problems manually. Furthermore, the control of the system is being dampened over time. This is suspicious, as it might mean this dampening approach was added post hoc because of struggling results. It should be noted that the adjoint PDE is not being solved in their cost function, and thus the respective adjoint formulas are not present in said cost function. This brings us to the the main contributions of this paper.

What limitations of prior work would be addressed by your new approach? The proposed framework in this paper goes beyond the framework in [15–17], because Control PINNs do not rely on data from the controller. Moreover, instead of evaluating PINNs in the "contex of the mature field of PDE-constrained optimization," [18] this paper proposes a framework that can be considered a new taxonomical entity within the genus of PINNs. The main contribution of Control PINNs is that an unknown controller u can be solved at the same time as the complex dynamical state of y.

How would you evaluate your new approach? This novel approach will be evaluated in three different ways. Firstly, an architect will be designed and then trained as a neural network while monitoring the loss function. Secondly, comparison of model results and reference solutions will be carried out for a problem with a known analytical solution. Thirdly, for problems with no known solution, the model results will be compared to simulation results using numerical analysis techniques. In other words, offline data generation for the controller can be collected for varying values of t and x (time and space). Then, this high precision data set can be plotted and validated with the solution found using a Mathematica solver.

Explain the key intuition behind your approach and why you think it is likely to be effective. Deep reinforcement learning (DRL) involves a feedback loop between an agent and an environment. The agent performs an action in the environment, which prompts a new updated state of the agent, and sometimes a given reward. This feedback loop forms the basis of an agent's policy. Correct policies that offer a reward for an agent's action strengthen the neural connectivity in certain action-state pathways [24]. By imparting known physical properties into the agent's neural network, you can supercharge these pathways to more efficiently reach a state of convergence.

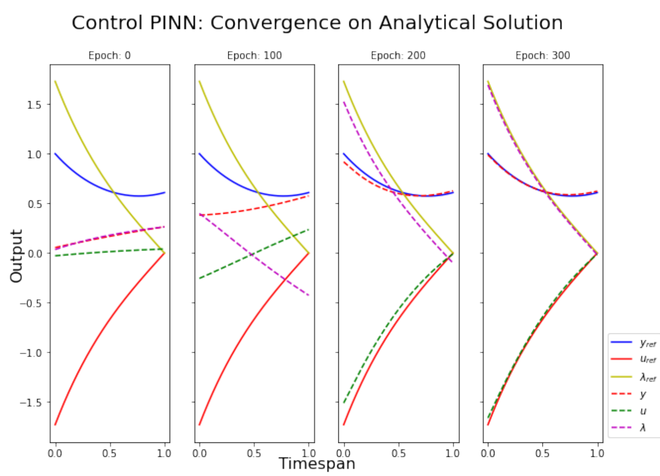

Figure 1: Results of the analytical problem. The optimal solutions of the system state, controller, and system push back on the controller are denoted by y_{ref}, u_{ref}, and λ_{ref}. Their respective predicted solutions found by the Control PINN model are denoted by y, u, and λ.

Reinforcement learning (RL) is the intersection of control theory and machine learning. As a result, RL learns control strategies to interact with a complex environment [24]. RL has proven itself successful in tackling fluid mechanic problems [25–30]. Intuitively, if both RL and PINNs are successful in the field of SciML, then combining them offers an interesting avenue for research. Control PINNs can be thought of as Deep Model Predictive Control, in the sense that they can solve difficult optimal nonlinear problems [24]. With this in mind, for a given problem Q-learning can learn a policy based on the Control PINN. In other words, the quality of the both the controller, and state of the system, provided by the Control PINN, can be systematized such that the best quality of action-state can be carried out. Before leveraging a Control PINN as an agent in a DRL framework, it first must be demonstrated that it can solve optimal/open loop control problems. The methodology in section 3 lays the ground work for Control PINNs to solve an analytical toy problem.

3 Methodology

We denote by $y(t)$ the solution of the ODE, and by $u(t)$ the control function. We seek to solve the control problem [31]:

$$\begin{aligned} &\min_{u(t)} \Psi(y) = \int_0^{t_f} g\big(y(t), u(t)\big)dt + w\big(y(t_f)\big) \\ &\text{subject to } \; y' = f(y,u), \; \forall t \in [0, t_f], \;\; y(0) = y_0. \end{aligned} \tag{1}$$

The optimality conditions are:

$$y'(t) = f\big(y(t), u(t)\big), \; \forall t \in [0, t_f]; \;\; y(0) = y_0^*; \tag{2a}$$

$$\begin{aligned} \lambda'(t) &= -f_y^T\big(y(t), u(t)\big)\,\lambda(t) - g_y^T\big(y(t), u(t)\big), \; \forall t \in [t_f, 0]; \\ \lambda(t_f) &= w_y^T\big(y(t_f)\big); \end{aligned} \tag{2b}$$

$$0 = -f_u\big(y(t), u(t)\big)\,\lambda(t) - g_u^T\big(y(t), u(t)\big), \; \forall t \in [0, t_f]. \tag{2c}$$

The process of solving the control problem detailed in eq. (1) is outlined in algorithm 2. The last three terms in eq. (3) can be thought of respectively as the system state, the system controller, and the push back on the controller by the system. In the context of autonomous vehicles, the system state can be thought of as the velocity and direction of the vehicle. By the same token, the system controller would be the software that governs the steering wheel and speed. Likewise, the system push back would be the feedback of the vehicle in response to the software's choices of direction and speed (e.g. the vehicle's shocks and brake pads).

What are the technical challenges with solving this problem? The main technical challenge involves learning the state of a dynamical system while at the same time finding its optimal control. Moreover, there is a tension between enforcing the boundary conditions and adhering to the constraints imposed by the physical laws. This is addressed by a scaling term in the loss function. Furthermore, as Control PINN is applied to increasingly more complex problems, some times a known unique solution may not be available. Such a solution would mainly be used for validation purposes.

Algorithm 2: The procedure to train a Control PINN model

Result: Training of a Control PINN that learns the optimal solution and the optimal control function for the given problem in (1)

1. Construct a network with inputs t, x (time and space), and outputs y, u, and λ (system state, system control, and system push back on control
2. Via auto differentiation and back-propagation, compute the following derivatives of the output w.r.t the input: $\frac{\delta y}{\delta x}, \frac{\delta y}{\delta \delta x}, \frac{\delta f}{\delta y}, \frac{\delta y}{\delta t}, \frac{\delta \lambda}{\delta t}, \frac{\delta f}{\delta u}, \frac{\delta g}{\delta y}, \frac{\delta g}{\delta u}$
3. With snapshots of the exact solution denoted by $y^*(t)$, minimize the loss function:

$$\begin{aligned} L = &\sum_i (t_{i+1} - t_i)\,\|y(t_i) - y^*(t_i)\|^2 \\ &+ \sum_i (t_{i+1} - t_i)\,\|D_t y(t_i) - f\big(y(t_i), u(t_i)\big)\|^2 \\ &+ \sum_i (t_{i+1} - t_i)\,\|D_t \lambda(t_i) + f_y^T\big(y(t_i), u(t_i)\big)\,\lambda(t_i) + g_y^T\big(y(t_i), u(t_i)\big)\|^2 \\ &+ \sum_i (t_{i+1} - t_i)\,\|f_u\big(y(t_i), u(t_i)\big)\,\lambda(t_i) + g_u^T\big(y(t_i), u(t_i)\big)\|^2. \end{aligned} \tag{3}$$

A visual representation of the Control PINN Architecture is found in fig. 2. Adaptive moment estimation (Adam) is used as the optimizer. The activation function of Exponential Linear Unit (ELU) is used. The neural density is 100 neurons per layer. There are ten hidden layers that proceed the input layer that takes in time (t) and space (x). The information of the system state (y) is passed to the controller (u) both directly and indirectly by a skip connection and several hidden layers, respectively. The aggregate information of the system state (y) and controller (u) is handled similarly in the context of the system's push back on the controller (λ). This architecture enables for the automatic differentiation of second order and mixed derivatives. This is necessary to impose the custom loss function detailed in algorithm 2.

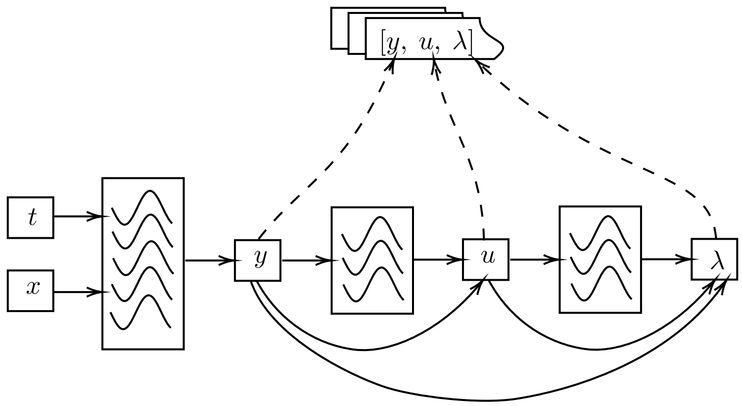

Figure 2: Visual Representation of Control PINN Architecture

4 Analytical Toy Problem

As a proof of concept, and to provide a foundation for methodology validation, let us consider the below test problem from [32]:

$$t_f = 1, \quad w\big(y(t_f)\big) = 0, \quad y(0) = y_0 = 1 \tag{4a}$$

$$f\big(y(t), u(t)\big) = \frac{1}{2}y(t) + u(t), \quad f_y\big(y(t), u(t)\big) = \frac{1}{2}, \quad f_u\big(y(t), u(t)\big) = 1 \tag{4b}$$

$$g\big(y(t), u(t)\big) = y^2(t) + \frac{1}{2}\,u^2(t), \quad g_y\big(y(t), u(t)\big) = 2\,y(t), \quad g_u\big(y(t), u(t)\big) = u(t) \tag{4c}$$

$$\lambda'(t) = -\frac{1}{2}\,\lambda(t) - 2y(t), \ \forall t \in [0,1]; \ \ \lambda(t_f) = 0 \tag{4d}$$

$$0 = -\lambda(t) - u(t), \ \forall t \in [0,1] \tag{4e}$$

The optimal solution is: $y^*(t) = \frac{2e^{3t}+e^3}{e^{3t/2}(2+e^3)}, \quad u^*(t) = \frac{2(e^{3t}-e^3)}{e^{3t/2}(2+e^3)}, \quad \lambda^*(t) = -u^*(t)$. We can then check that the analytical solution found by the Control PINN satisfies the optimality equations listed above. After a couple hundred epochs, the Control PINN converges on the optimal solution. This is shown in fig. 1, wherein y_{ref}, u_{ref}, and λ_{ref} respectively represent the reference solution for the system state y, the reference solution for the system controller u, and the reference solution of the system push back on the controller λ. Please refer to the following hyperlink to see an animation of the Control PINN model converging on the optimal solution.

5 Future Steps

Now that Control PINNs have been demonstrated as a proof-of-concept with an analytical toy problem, the next steps would involve increasing the problem complexity. For example, please refer to the following hyperlinks to see preliminary results for both a 1-dimensional heat problem, and a 2-dimensional spatio temporal predator-prey model (reaction diffusion). From there, Control PINNs can be implemented as a closed loop problem in the form of Navier Stokes flow control, or COVID-19 contact tracing. Starting with the most simple problem, and then adding more complex building blocks, offers a road map for leveraging Control PINNs as an agent for Deep Reinforcement Learning (DRL). For example, the results of the permutation invariant cart-pole swing up problem in [33] could then be compared to a Control PINN DRL implementation.

References

[1] Paul J. Atzberger. Importance of the mathematical foundations of machine learning methods for scientific and engineering applications, 2018.

[2] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021. doi: 10.1137/19M1274067.

[3] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems, 2021.

[4] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[5] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8):11618, Apr 2020. ISSN 1094-4087. doi: 10.1364/oe.384875. URL http://dx.doi.org/10.1364/OE.384875.

[6] QiZhi He, David Barajas-Solano, Guzel Tartakovsky, and Alexandre M. Tartakovsky. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141:103610, Jul 2020. ISSN 0309-1708. doi: 10.1016/j.advwatres.2020.103610. URL http://dx.doi.org/10.1016/j.advwatres.2020.103610.

[7] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[8] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[9] Maziar Raissi, Alireza Yazdani, and George Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367:eaaw4741, 01 2020. doi: 10.1126/science.aaw4741.

[10] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, Jan 2019. ISSN 1095-7197. doi: 10.1137/18m1229845. URL http://dx.doi.org/10.1137/18M1229845.

[11] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, Nov 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.07.048. URL http://dx.doi.org/10.1016/j.jcp.2019.07.048.

[12] Liu Yang, Dongkun Zhang, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations, 2018.

[13] Mohammad Amin Nabian and Hadi Meidani. A deep learning solution approach for high-dimensional random differential equations. *Probabilistic Engineering Mechanics*, 57:14–25, Jul 2019. ISSN 0266-8920. doi: 10.1016/j.probengmech.2019.05.001. URL http://dx.doi.org/10.1016/j.probengmech.2019.05.001.

[14] Dongkun Zhang, Ling Guo, and George Em Karniadakis. Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks, 2019.

[15] Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving pde-constrained control problems using operator learning, 2021.

[16] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach, 2019.

[17] Eric Aislan Antonelo, Eduardo Camponogara, Laio Oriel Seman, Eduardo Rehbein de Souza, Jean P. Jordanou, and Jomi F. Hubner. Physics-informed neural nets for control of dynamical systems, 2021.

[18] Saviz Mowlavi and Saleh Nabi. Optimal control of pdes using physics-informed neural networks, 2021.

[19] Salvatore Cuomo, Vincenzo Schiano di Cola, Fabio Giampaolo, Gianluigi Rozza, Maizar Raissi, and Francesco Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what's next, 2022.

[20] Ernst Hairer and Gerhard Wanner. Solving ordinary differential equations ii: Stiff and differential-algebraic problems. 2002.

[21] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. doi: 10.1109/MSP.2017.2743240.

[22] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019. doi: 10.1109/ACCESS.2019.2912200.

[23] Rahul Nellikkath and Spyros Chatzivasileiadis. Physics-informed neural networks for minimising worst-case violations in dc optimal power flow, 2021.

[24] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. doi: 10.1017/9781108380690.

[25] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508, 2020. doi: 10.1146/annurev-fluid-010719-060214. URL https://doi.org/10.1146/annurev-fluid-010719-060214.

[26] Siddhartha Verma, Guido Novati, and Petros Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1800923115. URL https://www.pnas.org/content/115/23/5849.

[27] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. Automating turbulence modeling by multi-agent reinforcement learning, 2020.

[28] Paul Garnier, Jonathan Viquerat, Jean Rabault, Aurélien Larcher, Alexander Kuhnle, and Elie Hachem. A review on deep reinforcement learning for fluid mechanics, 2021.

[29] Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, 2019. doi: 10.1017/jfm.2019.62.

[30] Dixia Fan, Liu Yang, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences*, 117(42):26091–26098, 2020. ISSN 0027-8424. doi: 10.1073/pnas.2004939117. URL https://www.pnas.org/content/117/42/26091.

[31] Jostein Barry-Straume, Arash Sarshar, Andrey A. Popov, and Adrian Sandu. Deep learning control of physics-informed neural networks for fast actuation in uncertain environments, 2021.

[32] William W. Hager. Runge-kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87:247–282, 2000.

[33] Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning, 2021.