Optimal Navigation in Microfluidics via the Optimization of a Discrete Loss

Petr Karnakov,[*] Lucas Amoudruz,[*] and Petros Koumoutsakos[†]
Computational Science and Engineering Laboratory, Harvard John A. Paulson School of Engineering and Applied Sciences, Cambridge, Massachusetts 02138, USA

(Received 22 August 2024; revised 24 November 2024; accepted 23 December 2024; published 29 January 2025; corrected 08 April 2025)

Optimal path planning and control of microscopic devices navigating in fluid environments is essential for applications ranging from targeted drug delivery to environmental monitoring. These tasks are challenging due to the complexity of microdevice-flow interactions. We introduce a closed-loop control method that optimizes a discrete loss (ODIL) in terms of dynamics and path objectives. In comparison with reinforcement learning, ODIL is more robust, up to 3 orders faster, and excels in high-dimensional action and state spaces, making it a powerful tool for navigating complex flow environments.

DOI: 10.1103/PhysRevLett.134.044001

The navigation and control of microfluidic devices are central in numerous fields, including precision medicine applications [1], micromanipulation [2–4], targeted drug delivery [5,6], and biomedical applications [7]. Nevertheless, there are numerous challenges in ensuring precision and efficiency of transport in highly viscous flows [8]. In this Letter, we introduce a novel method for effective navigation and control in microfluidic environments that integrates the optimization of a discrete loss (ODIL) [9] with neural networks (NNs) and automatic differentiation techniques. ODIL relies on casting the discrete form of the governing equations as a minimization problem. Here, ODIL builds upon the direct collocation method, a widely used technique in the optimal control community [10,11], but uses NNs to represent the policy. The use of NNs in the direct collocation method is not common due to the high computational cost associated with casting navigation as a constrained optimization problem. Indeed, these methods rely on numerical linear algebra with matrix decomposition [12], which makes the method expensive as the number of parameters increases.

The use of NNs in optimal control has been explored since the 1990s [13,14]. Recent advances include applications to nonlinear control [15,16] and model predictive control [17,18], employing differentiable control policies and leveraging modern computational tools like automatic differentiation (AD) [19,20]. The adjoint approach for nonlinear optimal control [21] has been suggested in the context of neural ordinary differential equations (ODEs) [22–24]. However, despite advances, there are limitations in differentiable control that hinder its effectiveness in solving flow navigation problems. The target information enters the objective function but it is not tightly coupled with the system dynamics which can lead to suboptimal solutions. Moreover, the deployed explicit integrators may lead to instabilities, further challenging the controllers. We also note that differentiable solvers and adjoint-based techniques assume that the forward problem is well posed. This may not always be the case as evidenced by the "notorious test problem" in optimal control [12,25], where the forward solution is unstable, but adding the knowledge of the target position regularizes the problem.

Sampling-based algorithms [26,27] for optimal motion planning offer a powerful approach for complex robotic tasks, particularly in high-dimensional or complex environments. However, they are limited by slow convergence to optimal solutions, potential difficulties in narrow passages, and sensitivity to parameters and are thus not well suited for complex environment and real-time or near-optimal solutions. To the best of our knowledge, these methods have not found any applications in fluid flows, except for recent work [28] that combines adaptive sampling methods with physics informed NNs.

Reinforcement learning (RL) [29] is gaining attention as a potent approach to tackle navigation problems in mobile robots [30] and for modeling and controlling complex fluid flows [6,31–39]. RL uses neural parametrization of policies within Markov decision processes and has shown promising results, especially with the integration of deep learning techniques [32,40,41]. Furthermore RL produces robust policies due to its closed-loop formulation [37,42,43]. More recently, extensions of RL to high-dimensional problems have shown great promise in solving complex tasks [44,45]. However, RL may involve extensive tuning of hyperparameters and heuristics in reward shaping and often results in computationally costly implementations.

The proposed ODIL framework [9] addresses several of the challenges mentioned above by leveraging automatic differentiation and standard gradient-based optimization methods [46], available in many modern machine learning

[*]These authors contributed equally to this work.
[†]Contact author: petros@seas.harvard.edu

packages. Moreover, it deploys a custom multigrid technique for accelerated convergence [47], offering a fast and convenient tool to solve navigation and control problems for microfluidics. We demonstrate the effectiveness of the ODIL approach on a series of benchmark navigation and control problems in microscale flow environments. In comparison to RL, which currently represents the state of the art for path planning and control problems in these applications [6,34–37,48,49], ODIL is more robust, in particular, for high-dimensional state and action spaces, while requiring between 1 and 3 orders of magnitude fewer policy evaluations.

Methods—We consider path planning problems described by optimization of the form

$$\text{minimize}\, T \tag{1}$$

$$\text{subject to}\, \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{a}(\mathbf{x})), \qquad t \in (0, T), \tag{2}$$

$$\mathbf{x}(0) = \mathbf{x}_{\text{start}}, \qquad \mathbf{x}(T) = \mathbf{x}_{\text{target}}, \tag{3}$$

where $\mathbf{x}(t)$ represents the state of the system, $\mathbf{a}$ is the control policy, $\mathbf{f}$ describes the dynamics of the system, T is the travel time, $\mathbf{x}_{\text{start}}$ is the initial state, and $\mathbf{x}_{\text{target}}$ is the target state. The task is to find a policy $\mathbf{a} \in A$ that minimizes the travel time T under the constraints [Eqs. (2) and (3)], where A is a case-dependent feasible set of policies. We represent the policy with a fully connected NN with weights θ, $\mathbf{a} = \mathbf{a}_\theta$. The NN consists of two hidden layers of size 128×128, tanh activation in the hidden layers, and a case-specific activation function in the output layer. For brevity, we assume that the right-hand side $\mathbf{f}$ and the policy $\mathbf{a}_\theta$ do not explicitly depend on time t.

ODIL approach—The ODIL approach reduces the constrained minimization problem [Eqs. (1)–(3)] to unconstrained minimization of the loss function

$$\mathcal{L}(\mathbf{x}, \theta) = \sum_{n=0}^{N-2} \left| \Delta \mathbf{x}^{n+1/2} - \mathbf{f}_\theta^{n+1/2} \Delta t \right|^2 + \lambda T, \tag{4}$$

with respect to the NN parameters θ and the trajectory $\mathbf{x}$ discretized on a uniform grid of N points in time, where $\lambda > 0$ is a penalization factor. In Eq. (4), we used the notation

$$\begin{aligned} \Delta \mathbf{x}^{n+1/2} &= \mathbf{x}^{n+1} - \mathbf{x}^n, \\ \mathbf{f}_\theta^{n+1/2} &= \mathbf{f}(\mathbf{x}^{n+1/2}, \mathbf{a}_\theta(\mathbf{x}^{n+1/2})), \\ \mathbf{x}^{n+1/2} &= \frac{\mathbf{x}^{n+1} + \mathbf{x}^n}{2}, \\ \Delta t &= \frac{\Delta \mathbf{x} \cdot \mathbf{f}_\theta}{\mathbf{f}_\theta \cdot \mathbf{f}_\theta}, \qquad T = (N-1)\Delta t, \end{aligned}$$

where $n = 0, \ldots, N-2$, and the scalar product defined as $\mathbf{u} \cdot \mathbf{v} = \sum_{n=0}^{N-2} \mathbf{u}^{n+1/2} \cdot \mathbf{v}^{n+1/2}$ for vectors $\mathbf{u}$ and $\mathbf{v}$ of length N. This loss function contains the residual of the governing ODE [Eq. (2)] discretized using the midpoint rule on a grid of N points in time. Unless stated otherwise, the initial and final conditions are imposed exactly by setting $\mathbf{x}^0 = \mathbf{x}_{\text{start}}$ and $\mathbf{x}^{N-1} = \mathbf{x}_{\text{target}}$, and the initial guess for the trajectory is a straight line connecting $\mathbf{x}_{\text{start}}$ and $\mathbf{x}_{\text{target}}$. The optimization process relies on AD and the Adam optimization technique. Further details of the optimization are described in Supplemental Material [50], which includes Refs. [51–58].

RL approach—The system is controlled by an agent that determines the appropriate action based on the current state at regular time intervals $\tau > \Delta t$. The agent's control value $\mathbf{a}$ remains constant during each time interval. When it comes to navigation problems, the objective is to reach the target within a close proximity $\delta > 0$ in the shortest possible time. To represent this objective, a negative reward is assigned after each action. Furthermore, to aid the system in reaching the target, a reward shaping term is utilized. This term yields a positive reward when the system's position moves closer to the target following an action [59]. Consequently, the reward obtained after the tth action can be expressed as follows:

$$r_t = -\kappa\tau + |\mathbf{x}_t - \mathbf{x}_{\text{target}}| - |\mathbf{x}_{t+1} - \mathbf{x}_{\text{target}}|, \tag{5}$$

where $\kappa > 0$ is denoted as the time penalty coefficient. We note that the shaping term in Eq. (5) does not modify the objective as it is derived from a potential [59]. An episode is terminated if the time exceeds a value $T_{\max}$ or if the system's position has reached the target within the distance δ. Details about the RL training are listed in Supplemental Material [50].

Magnetic swimmers—Artificial bacterial flagella (ABFs) are microswimmers that are propelled through external magnetic fields in viscous flow environments [60]. When these swimmers are immersed in a rotating, uniform magnetic field, they exhibit a magnetic moment, which induces a torque along their primary axis. The helicoidal shape of ABFs causes the torque to be converted into linear velocity in the direction perpendicular to the plane of rotation of the external magnetic field, denoted as $\mathbf{p}$. The average velocity magnitude of an ABF is given by [61,62]

$$v(\omega; b, \omega_c) = \begin{cases} b\omega, & \text{if } \omega \le \omega_c, \\ b(\omega - \sqrt{\omega^2 - \omega_c^2}), & \text{otherwise,} \end{cases}$$

where ω is the angular frequency of the magnetic field and the parameters b and ω_c depend on the shape and magnetic properties of the ABF [61]. Despite a uniform magnetic field, it is possible to control multiple ABFs independently if their shapes differ [6]. We describe the above system in two dimensions and neglect hydrodynamic and magnetic interactions between swimmers. The resulting system of ODEs reads $\dot{\mathbf{x}}_i = v(\omega; b_i, \omega_{c,i})\mathbf{p}$, $i = 2, \ldots, M$, where M is

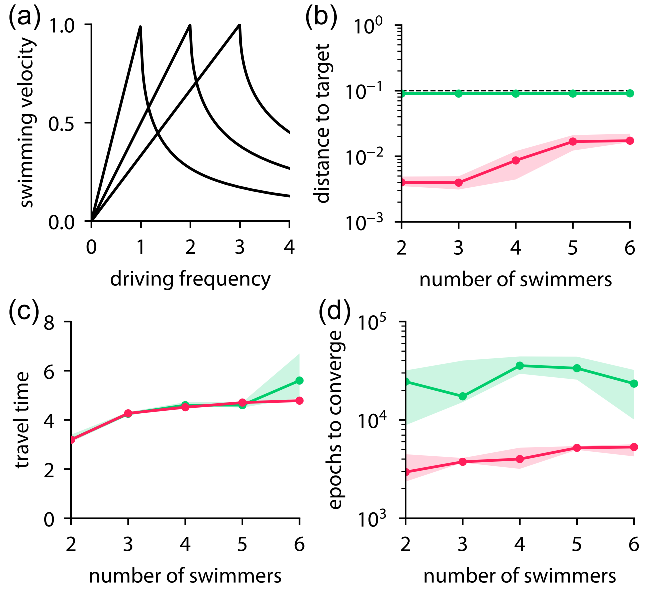

FIG. 1. Magnetic swimmers. (a) Velocity response of three swimmers depending on the driving frequency. (b)–(d) Distance to target, travel time, and number of epochs until convergence versus the number of swimmers for ODIL (red line) and RL (green line). Shades are the 20th to 80th percentiles, solid lines are the medians over 10 realizations. The dashed line shows the threshold distance to target δ.

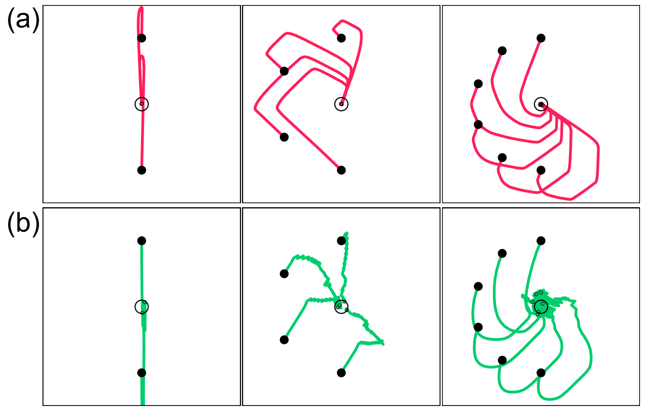

FIG. 2. Trajectories of $M = 2$, 4, and 6 magnetic swimmers with starting positions • and target ○ using (a) ODIL (red line) and (b) RL (green line).

the number of swimmers. The task consists in guiding all M swimmers from their initial position $\mathbf{x}_{\mathrm{start},i}$ toward a target $\mathbf{x}_{\mathrm{target}}$ in the least amount of time, by modulating the magnetic field direction $\mathbf{p}$ and angular frequency ω. The direction of the magnetic field is set to $\mathbf{p} = \mathbf{a}_p/|\mathbf{a}_p|$, with $\mathbf{a}_p = (a_{\theta,1}, a_{\theta,2})$. the third component of $\mathbf{a}_\theta$ encodes the angular frequency ω. We set $\omega_{c,i} = i$ and $b_i = 1/i$, $i = 1, 2, \ldots, M$ (Fig. 1). The target of each ABF is placed at the origin and the starting positions are located on the left side of the unit circle, $\mathbf{x}_{\mathrm{start},i} = (\cos\phi_i, \sin\phi_i)$, with $\phi_i = \pi/2 + i\pi/(M-1)$, $i = 1, 2, \ldots, M$. See Supplemental Material [50] for details about the ODIL and RL training parameters.

Figure 2 shows the trajectories of ABFs that follow policies obtained with the two approaches. Both methods successfully steer all M swimmers for $M \in \{2, 4, 6\}$. The trajectories learned with the two methods are different, but we note that this problem has many solutions [6]. ODIL converges an order of magnitude faster than RL in terms of number of epochs (Fig. 1). Both methods seem to find a similar travel time, but the RL approach degrades when the number of swimmers is larger than 5 compared to ODIL. Furthermore ODIL brings the swimmers closer to the target than RL, which stops when all swimmers reach the target within the distance $\delta = 0.1$.

As the number of ABFs increases, this problem becomes increasingly challenging, especially when the parameters $\omega_{c,i}$ are not well separated. Furthermore, with a large number of swimmers, hydrodynamic interactions become significant, increasing the difficulty to control ABF swarms. Addressing the control of such swarms will be the focus of further research.

Transport with vortices—Here we consider a system of multiple particles transported through a grid of vortices [3,4]. The task is to bring the particles from their starting positions to individual targets by controlling the intensity of each vortex. The system consists of M particles passively transported by the vortical flow as $\dot{\mathbf{x}}_i = \mathbf{u}(\mathbf{x}_i)$. The velocity field is a superposition of $M \times M$ vortices

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{M\times M} \omega_i \mathbf{u}_V(\mathbf{x} - \mathbf{c}_i),$$

with intensities $\omega_i \in (-1, 1)$ and centers $\mathbf{c}_i$ forming a uniform grid in $[0.5, M-0.5] \times [0.5, M-0.5]$, and each vortex produces a flow proportional to $\mathbf{u}_V(x, y) = e^{-(x^2+y^2)/0.72}(-y, x)$. The task is to transport the particles from their starting positions $\mathbf{x}_{\mathrm{start},i} = (0.5 + i, 0)$ to targets $\mathbf{x}_{\mathrm{target},i} = (0.5 + i, M)$ in the minimal time T by controlling the vortex intensities ω_i as a function of the current positions of all beads. See Supplemental Material [50] for details about the ODIL and RL parameters.

Figure 3 compares the control policies of ODIL and RL for $M = 2, 3, \ldots, 8$ swimmers. Example trajectories obtained with ODIL are shown in Fig. 4. See Supplemental Material [50] for trajectories obtained with RL and for other values of M. For $M = 2$, both methods find similar trajectories in all realizations. For $M = 3$, only 30% of realizations of RL lead to a valid path to the targets, and the successful trajectories appear more erratic than those found by ODIL. For larger values $M \geq 4$, the RL does not produce any valid policy, while ODIL successfully solves the problem. ODIL tends to gather the particles in a central corridor before dispatching each particle to their respective targets. Since the dimension of the action space grows

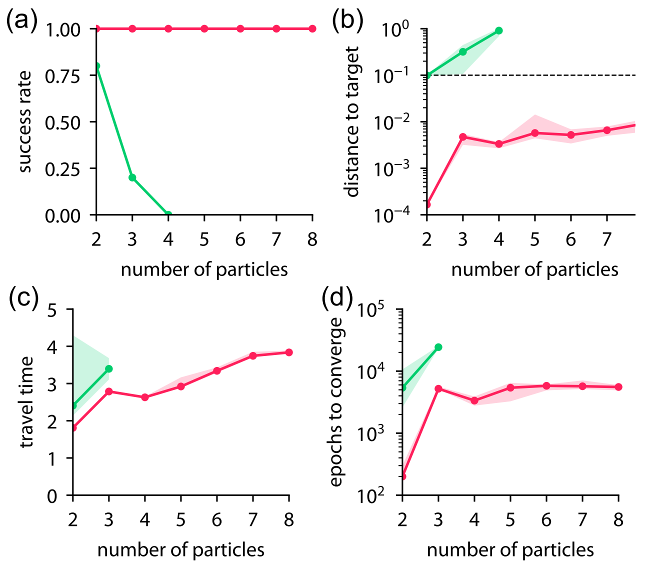

FIG. 3. Manipulation by vortices. Performance of ODIL (red line) and RL (green line) depending on the number of particles: (a) success rate, (b) distance to target, (c) number of epochs until convergence, and (d) travel time. Shades are the 20th–80th percentiles, solid lines are the medians over ten realizations. The dashed line shows the threshold distance to target δ.

quadratically with M, the use of gradients by ODIL becomes advantageous over RL for finding valid paths.

Computational cost—We compare the cost of ODIL against RL through two metrics. The first metric is the number of forward and backward passes through the NN until convergence and is relevant when solving the dynamics of the system is relatively cheap. The second metric is wall time until convergence, which reflects both the backpropagation through the NN and solving the dynamics of the environment. The RL approach uses a replay memory buffer to store past experiences, used in minibatches of size B to estimate the off-policy objective loss. Each training step thus corresponds to $B + 1$ forward passes and one backward pass through the NN. In this study we used one training step per experience and $B = 256$. The ODIL approach, on the other hand, requires one forward and one backward pass per discretization point along one trajectory. In all cases, we define that the training is converged when the travel time is within 1% of the best value.

We report these metrics for the two cases detailed in this Letter with different values of M. Additional examples given in Supplemental Material [50] show similar trends. It is clear from Table I that,for all benchmark problems, the computational cost is orders of magnitude lower with ODIL compared to RL. While the number of training epochs was similar for both methods, the RL approach requires one to evaluate the policy over minibatches at each policy update, explaining the large cost associated with this approach. This resulted in run times larger by factors of 10–100 for the RL approach over ODIL. We remark that the RL method does not always converge to a satisfactory solution within a reasonable time, and the numbers reported in Table I are computed among the successful trials, when available. We expect that ODIL remains advantageous even for more costly dynamical systems. Indeed, in addition to the faster convergence of ODIL over RL, ODIL is more robust in finding satisfactory solutions at higher state and action spaces.

We note that RL and ODIL operate under fundamentally different assumptions about system dynamics. ODIL, a model-based approach, requires a differentiable ODE model of system dynamics, whereas model-free RL relies on substantial environment interaction to converge, which may explain their performance differences. In cases of incomplete system knowledge, ODIL could potentially

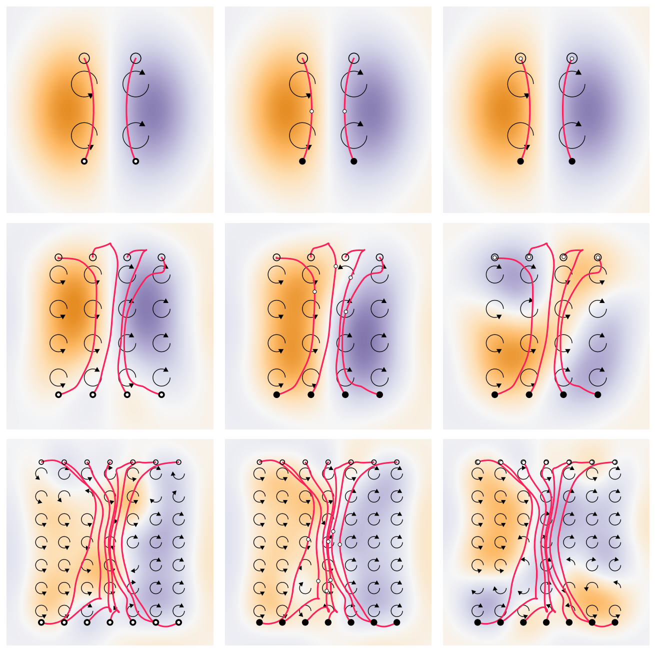

FIG. 4. Manipulation by vortices using ODIL (red line) for $M = 2$, 4, and 7. Each row corresponds to one trajectory at times $t/T = 0$, 0.5, and 1. Vorticity field with arrows showing the intensity and orientation of vortices.

TABLE I. Number of policy evaluations and total wall time for training the policy with ODIL and RL (median over ten random seeds).

	Policy evaluations		Wall time (s)	
Case	ODIL	RL	ODIL	RL
ABFs, $M = 3$	$1.29 + 10^6$	5.47×10^8	1.68×10^2	1.55×10^4
ABFs, $M = 5$	1.29×10^6	8.44×10^8	1.75×10^2	2.45×10^4
Vortices, $M = 2$	1.29×10^6	1.35×10^8	9.90×10^1	4.72×10^3
Vortices, $M = 3$	1.29×10^6	6.18×10^8	1.04×10^2	2.54×10^4
Vortices, $M = 8$	1.29×10^6	⋯	1.44×10^2	⋯

integrate a data-driven model of unknown dynamics, which will be explored in future work. Finally, while this Letter uses a single initial condition, Supplemental Material [50] includes two examples demonstrating closed-loop control with multiple initial conditions, enhancing resilience to external noise.

Summary—We have presented the ODIL method to solve closed-loop control and navigation problems described by ODEs. The formulation of the method relies on the optimization of a loss function that combines the residuals of the governing equations, the initial and target positions of the system, and the objective. We perform the optimization with standard machine learning tools that allows us to seamlessly describe the control policy with NNs. We have tested the new method on a variety of navigation problems and compared against a state-of-the-art RL algorithm. In all cases, ODIL has a lower computational cost than RL while producing policies with similar or better performance. Furthermore, we have shown that RL fails to reach the target when the action space is high dimensional, while ODIL reliably solves the problem in such cases.

ODIL is well suited for problems with known targets due to its implicit formulation and the use of gradients. On the other hand, RL requires reward shaping and sometimes extensive tuning to work correctly. Future applications of ODIL can consider more complex dynamical systems such as those described by partial differential equations.

Acknowledgments—Authors acknowledge the use of computing resources from the Swiss National Supercomputing Centre (CSCS) under Project No. s1160.

[1] D. Barbolosi, J. Ciccolini, B. Lacarelle, F. Barlési, and N. André, Nat. Rev. Clin. Oncol. **13**, 242 (2016).
[2] H. Yun, K. Kim, and W. G. Lee, Biofabrication **5**, 022001 (2013).
[3] Z. Ye, E. Diller, and M. Sitti, J. Appl. Phys. **112**, 064912 (2012).
[4] F. N. P. Basualdo, G. Gardi, W. Wang, S. O. Demir, A. Bolopion, M. Gauthier, P. Lambert, and M. Sitti, IEEE Rob. Autom. Lett. **7**, 2156 (2022).
[5] B. Liebchen and H. Löwen, Europhys. Lett. **127**, 34003 (2019).
[6] L. Amoudruz and P. Koumoutsakos, Adv. Intell. Syst. **4**, 2100183 (2022).
[7] A.-I. Bunea and R. Taboryski, Micromachines **11**, 1048 (2020).
[8] M. Safdar, J. Simmchen, and J. Jänis, Environ. Sci. **4**, 1602 (2017).
[9] P. Karnakov, S. Litvinov, and P. Koumoutsakos, Proc. Natl. Acad. Sci. Nexus **3**, 005 (2024).
[10] M. Kelly, SIAM Rev. **59**, 849 (2017).
[11] R. Bordalba, T. Schoels, L. Ros, J. M. Porta, and M. Diehl, IEEE Trans. Rob. **39**, 183 (2022).
[12] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* (SIAM, Philadelphia, 2010).
[13] F.-C. Chen, IEEE Control Syst. Mag. **10**, 44 (1990).
[14] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, Automatica **28**, 1083 (1992).
[15] S. Adhau, V. V. Naik, and S. Skogestad, in *Proceedings of the 2021 60th IEEE Conference on Decision and Control (CDC)* (IEEE, New York, 2021), pp. 295–300.
[16] W. Jin, Z. Wang, Z. Yang, and S. Mou, Adv. Neural Inf. Process. Syst. **33**, 7979 (2020), https://proceedings.neurips.cc/paper_files/paper/2020/hash/5a7b238ba0f6502e5d6be14424b20ded-Abstract.html.
[17] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, Adv. Neural Inf. Process. Syst. **31**, 8299 (2018), https://proceedings.neurips.cc/paper/2018/hash/ba6d843eb4251a4526ce65d1807a9309-Abstract.html.
[18] B. Karg and S. Lucia, .IEEE Trans. Syst. Man Cybern. **50**, 3866 (2020).
[19] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, J. Mach. Learn. Res. **18**, 1 (2018), https://www.jmlr.org/papers/v18/17-468.html.
[20] Y. Cao, Journal of Process Control **15**, 851 (2005).
[21] I. O. Sandoval, P. Petsagkourakis, and E. A. del Rio-Chanona, arXiv:2210.11245.
[22] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Adv. Neural Inf. Process. Syst. **31** (2018), https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.
[23] G.-H. Liu, T. Chen, and E. Theodorou, Adv. Neural Inf. Process. Syst. **34**, 25267 (2021), https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.
[24] A. Katrutsa, I. Oseledets, and S. Utyuzhnikov, arXiv:2406.01991.
[25] H. G. Bock, in *Numerical Treatment of Inverse Problems in Differential and Integral Equations: Proceedings of an International Workshop, Heidelberg, Fed. Rep. of Germany, 1982* (Springer, New York, 1983), pp. 95–121.
[26] S. Karaman and E. Frazzoli, Int. J. Robot. Res. **30**, 846 (2011).
[27] O. Arslan, K. Berntorp, and P. Tsiotras, in *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, New York, 2017), pp. 4991–4996.
[28] Y. Liu, L. Chen, Y. Chen, and J. Ding, Entropy **26**, 451 (2024).
[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 2018).
[30] K. Zhu and T. Zhang, Tsinghua Sci. Technol. **26**, 674 (2021).
[31] G. Reddy, J. Wong-Ng, A. Celani, T. J. Sejnowski, and M. Vergassola, Nature (London) **562**, 236 (2018).
[32] S. Verma, G. Novati, and P. Koumoutsakos, Proc. Natl. Acad. Sci. U.S.A. **115**, 5849 (2018).
[33] P. Gunnarson, I. Mandralis, G. Novati, P. Koumoutsakos, and J. O. Dabiri, Nat. Commun. **12**, 7143 (2021).
[34] S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale, Phys. Rev. Lett. **118**, 158004 (2017).
[35] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, and K. Gustavsson, Chaos **29**, 103138 (2019).

[36] M. Nasiri and B. Liebchen, New J. Phys. **24**, 073042 (2022).
[37] L. Amoudruz, S. Litvinov, and P. Koumoutsakos, arXiv:2404.02171.
[38] Y. Zhu, J.-H. Pang, and F.-B. Tian, Front. Phys. **10**, 870273 (2022).
[39] H. Li, Q. Zhang, and D. Zhao, IEEE Trans. Neural Networks Learn. Syst. **31**, 2064 (2019).
[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, arXiv:1312.5602.
[41] G. Novati and P. Koumoutsakos, in *International Conference on Machine Learning* (PMLR, Long Beach, California, 2019), pp. 4851–4860.
[42] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, IEEE Trans. Rob. **35**, 124 (2018).
[43] A. Zavoli and L. Federici, J. Guid. Control Dyn. **44**, 1440 (2021).
[44] M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal, Nat. Med. **24**, 1716 (2018).
[45] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, in *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (IEEE, New York, 2019), pp. 8248–8254.
[46] D. P. Kingma and J. Ba, arXiv:1412.6980.
[47] P. Karnakov, S. Litvinov, and P. Koumoutsakos, Eur. Phys. J. E **46**, 59 (2023).
[48] C. Mo, Q. Fu, and X. Bian, Phys. Rev. E **108**, 044408 (2023).
[49] C. Mo, G. Li, and X. Bian, Front. Phys. **11**, 1279883 (2023).
[50] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.134.044001 for details about the methods and a set of additional examples, which includes Refs. [51–58].
[51] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from https://www.tensorflow.org/.
[52] R. Frostig, M. J. Johnson, and C. Leary, Syst. Mach. Learn. **4** (2018), https://research.google/pubs/compiling-machine-learning-programs-via-high-level-tracing/.
[53] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid* (Elsevier, New York, 2000).
[54] S. M. Martin, D. Wälchli, G. Arampatzis, A. E. Economides, P. Karnakov, and P. Koumoutsakos, Comput. Methods Appl. Mech. Eng. **389**, 114264 (2022).
[55] P. J. Vach, P. Fratzl, S. Klumpp, and D. Faivre, Nano Lett. **15**, 7064 (2015).
[56] A. Najafi and R. Golestanian, Phys. Rev. E **69**, 062901 (2004).
[57] B. Hartl, M. Hübl, G. Kahl, and A. Zöttl, Proc. Natl. Acad. Sci. U.S.A. **118**, e2019683118 (2021).
[58] K. Huang, S. Lale, U. Rosolia, Y. Shi, and A. Anandkumar, arXiv:2112.07746.
[59] A. Y. Ng, D. Harada, and S. Russell, in *International Conference on Machine Learning* (Citeseer, 1999), Vol. 99, pp. 278–287.
[60] L. Zhang, J. J. Abbott, L. Dong, B. E. Kratochvil, D. Bell, and B. J. Nelson, Appl. Phys. Lett. **94**, 064107 (2009).
[61] D. Schamel, M. Pfeifer, J. G. Gibbs, B. Miksch, A. G. Mark, and P. Fischer, J. Am. Chem. Soc. **135**, 12353 (2013).
[62] P. J. Vach, N. Brun, M. Bennet, L. Bertinetti, M. Widdrat, J. Baumgartner, S. Klumpp, P. Fratzl, and D. Faivre, Nano Lett. **13**, 5373 (2013).

Correction: The previously published Fig. 4 contained errors and has been replaced.