https://doi.org/10.1038/s44182-024-00013-0

FALCON: Fourier Adaptive Learning and Control for Disturbance Rejection Under Extreme Turbulence

Sahin Lale[1,3], Peter I. Renn[1,3], Kamyar Azizzadenesheli[2], Babak Hassibi[1], Morteza Gharib[1] & Anima Anandkumar[1] ✉

Controlling aerodynamic forces in turbulent conditions is crucial for UAV operation. Traditional reactive methods often struggle due to unpredictable flow and sensor noise. We present FALCON (Fourier Adaptive Learning and Control), a model-based reinforcement learning framework for effective modeling and control of aerodynamic forces under turbulent flows. FALCON leverages two key insights: turbulent dynamics are well-modeled in the frequency domain, and most turbulent energy is concentrated in low-frequencies. FALCON learns a concise Fourier basis to model system dynamics from 35 s of flow data. To address sensor limitations, FALCON models dynamics using a short history of actions and measurements. With this approach, FALCON applies model predictive control for safe and efficient control. Tested in the Caltech wind tunnel under highly turbulent conditions, FALCON learns to control the underlying nonlinear dynamics with less than 9 min of data, consistently outperforming state-of-the-art methods. We provide guarantees for FALCON, ensuring stability and robustness.

Turbulent atmospheric winds often contain transient flow disturbances and aerodynamic forces that can affect a variety of systems and structures[1]. These forces are particularly significant for aerodynamic technologies like unmanned aerial vehicles (UAVs) and wind turbines, which rely on fluid interaction for regular operation and can be damaged when operating in turbulent conditions[2–4]. Developing active control strategies to mitigate the effects of these turbulent forces is one of the most important challenges in the safe deployment of UAV technologies or extending the lifetime and reliability of wind turbines[5–7]. Developing and utilizing complex flow models for control is challenging in real-time due to sensor noise, low-latency, and high-frequency control requirements[8–10]. For example, modeling the flow requires advanced computational fluid dynamics (CFD) solvers that are often too slow for real-time application and fail under noisy settings[11].

Conventional control strategies for UAVs, such as proportional-integral-derivative (PID) controllers, are designed to reactively correct inertial deviations from the desired trajectory without taking into consideration the underlying flow dynamics or the source of the disturbance[12–14]. These approaches are often insufficient for maintaining stability in extreme atmospheric turbulence, which prevents deployment of these technologies in safety-critical scenarios such as deploying unmanned aerial vehicles (UAV) in densely populated urban areas[15–17], such as Fig. 1A. A line of work proposes adding Dryden or von Karman turbulence models, a simplified side dynamics, as a correction to the base dynamical model[18–20]. Such corrections are inspired by first-order physics and introduce ordinary differential equations (ODE) correction to the UAV dynamics. These approaches are also often used in practice and come with the limitations that the correction is known, and an Euler method with fine time resolution is needed for the ODE correction.

In contrast, biological swimmers and flyers have the ability to directly observe and respond to the physics responsible for changes in motion[21–24]. By drawing inspiration from these biological systems, there have been considerable efforts to improve control strategies for UAVs by using easily measurable flow quantities, such as pressure, to anticipate and mitigate the effects of turbulent disturbances[25–30]. The majority of these works again utilize the flow-sensing information within PID control frameworks which limits their desirable performance to low velocities[25–28] or they consider uniform wind/flow scenarios where the eddies and gusts have smaller scale than the UAVs which result in small aerodynamic disturbances[29,30].

To tap the potential of flow-sensing in designing disturbance rejection policies, reinforcement learning (RL), a machine learning area, has been recognized as a promising framework, due to its ability to learn and adapt to the unmodeled dynamics and design nonlinear policies with various objectives. Most of the prior works on RL for flow control have focused on model-free RL techniques and developed in CFD simulations. Model-free

[1]California Institute of Technology, Pasadena, USA. [2]Nvidia Corporation, Santa Clara, CA, USA. [3]These authors contributed equally: Sahin Lale, Peter I. Renn. ✉e-mail: anima@caltech.edu

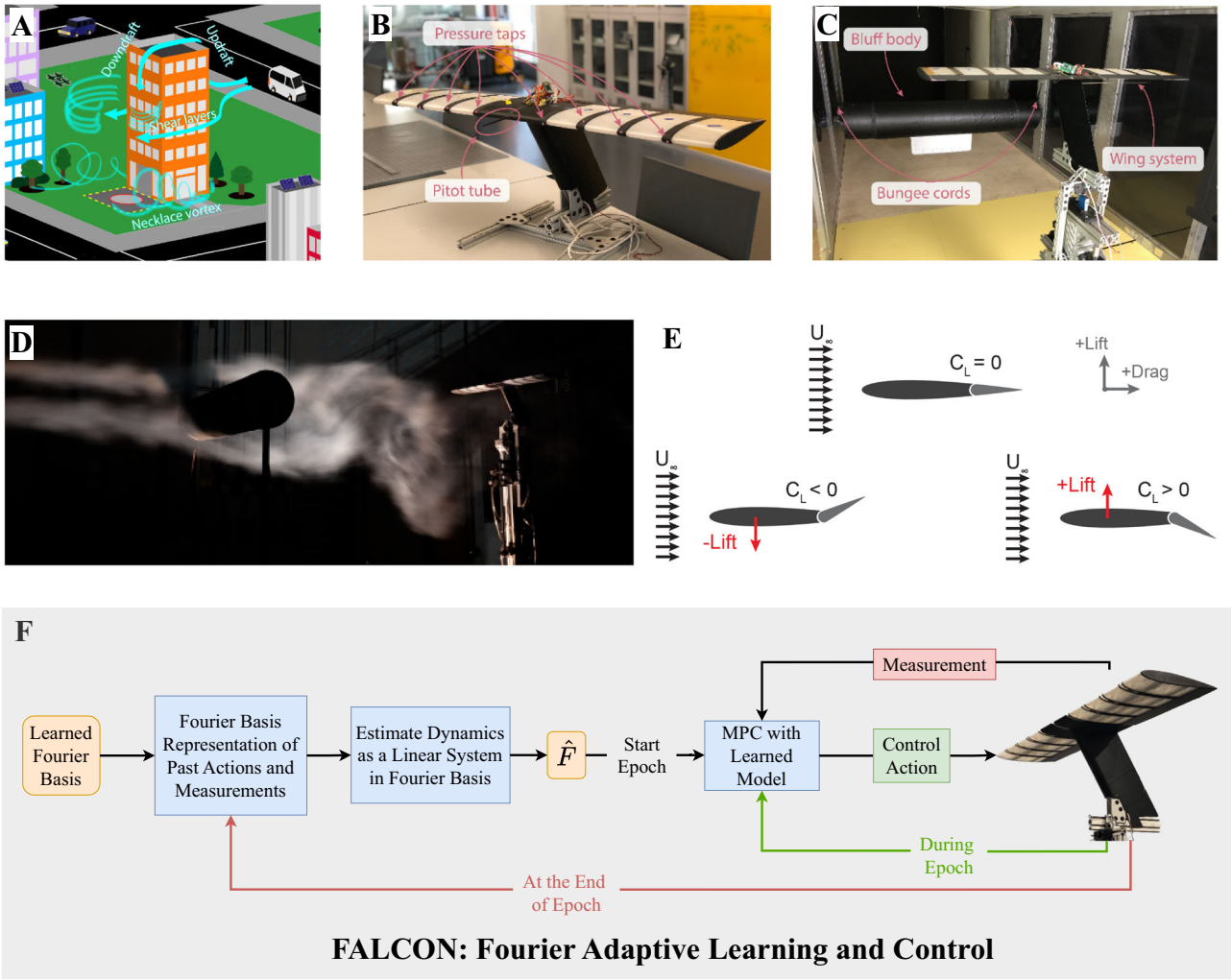

Fig. 1 | Problem and Experiment Setup. A Complex airflow structures in urban environments. **B** The wing has 9 sensors to measure the airflow (8 equally spaced pressure taps and 1 pitot tube) and is mounted on a one-dimensional load cell to measure the lift. Trailing-edge flaps change orientation to manipulate the aerodynamic forces. **C** Experiment setup to create irregular turbulent wake of a bluff body under high wind speeds. **D** Smoke visualization of the turbulent wake of a cylinder at a smaller Reynolds number. This image is obtained at the Caltech Real Weather Wind Tunnel system at a significantly lower flow speed than the experiments conducted in this work for visualization purposes. The actual flow conditions used in our studies were too turbulent to have clear smoke visualization. **E** Under a uniform flow U_∞, symmetric airfoils do not have any vertical aerodynamic forces on them when they are aligned with the airflow. However, altering the position of a trailing edge flap on the airfoil can modify the lift coefficient C_L, yielding an upward or downward aerodynamic lift force. **F** Outline of FALCON, a model-based reinforcement learning framework that allows effective modeling and control of the aerodynamic forces due to turbulent flow dynamics and achieves state-of-the-art disturbance rejection performance.

RL methods do not construct an explicit model of the system dynamics and aim to learn the control policies directly through interactions with the system[31]. Therefore, they are the most intuitive choices for policy design in environments difficult to model such as turbulent flow dynamics. Among these model-free RL works, Bieker et al.[32] introduced a novel framework with online learning to predict and control flow in a 2D CFD simulation. Gunnarson et al.[33] introduced an algorithm to navigate a simulated "swimmer" across an unsteady flow in a 2D simulation. In the experimental studies, Fan et al.[34] demonstrated the first experimental applications of model-free RL in fluid mechanics. Recently, Renn and Gharib[35] used a model-free RL method for controlling the aerodynamic forces on an airfoil under turbulent flow in an experimental setting (similar to the one considered in this work) and achieved state-of-the-art disturbance rejection performance, outperforming PID control. They also documented that the power spectrum of the turbulent flow at high Reynolds numbers is dominated by the low-frequency components which inspired the development of the algorithm in this work. However, despite this strong empirical performance, their method suffers from well-known limitations of the model-free RL methods, namely, extensive and laborious data collection, and brittle policies.

In this work, we take on the challenge of designing a model-based RL framework for flow-informed aerodynamic control in a highly turbulent and vortical environment to overcome these limitations. Unlike their model-free counterparts, model-based RL methods pursue joint objectives of model learning and policy optimization[36]. They are strong alternatives to model-free methods due to their sample efficiency, ability to adapt to changing conditions and to generalize to unseen conditions via the learned model. Moreover, most real-world systems are governed by physics, which can be incorporated into model learning. This also enables them to generalize better to out-of-distribution samples, which is crucial in safety-critical tasks. However, despite these promises, most of the current model-based RL methods are rarely implemented in real-world systems due to their need for highly accurate models and the challenges that partial observability brings.

In most real-world dynamical systems, the system state is hidden, and instead, the controlling agent observes a nonlinear and noisy measurement

of the state, e.g. through sensors. This partial observability brings uncertainties in modeling the system dynamics and designing the policies[37]. It also violates the common design assumption of the Markov property in the collected samples, which significantly complicates the modeling task[38]. These challenges make model-based RL remarkably difficult in real-world systems. To remedy these challenges, the majority of the model-based RL methods rely on the expressive power of deep neural networks (DNN) in modeling the dynamics. However, these approaches require a vast number of samples and usually yield black-box models. Thus, these methods are most relevant in stationary and safe environments such as robotic manipulations[39]. However, under unsteady conditions such as complex turbulent flow fields, we require efficient and adaptive modeling for generalizable learning.

In this article, we propose an efficient model-based RL algorithm, **F**ourier **A**daptive **L**earning and **Con**trol (FALCON), for online control of unknown partially observable nonlinear dynamical systems, in particular for disturbance rejection under extreme flow conditions for which the turbulence dynamics and future events are a priori unknown (Fig. 1F). FALCON leverages the domain knowledge that the underlying turbulent flow dynamics are well-modeled in the frequency domain and that most of the energy in the turbulent flows is present in low-frequency components[35,40]. Therefore, it learns the underlying partially observable system in a succinct Fourier Series basis. In short, FALCON constructs a highly selective and nonlinear feature representation of the system in which the dynamical evolution is approximately linear (in the feature space), resulting in an end-to-end highly nonlinear and accurate model of the underlying system dynamics. The Fourier coefficients provide good inductive bias to learn this cross-coupling automatically from observations. The soundness of model learning using constructed Fourier bases is theoretically guaranteed, and the algorithm design in this work is mainly towards low error in the next step prediction. In this work, we evaluate the accuracy and soundness of the learned model using controller performance equipped with the learned model, which is the quantity of interest in RL and control. The low error prediction error is sufficient to achieve the learning and control guarantee and arrive at the proposed practical algorithm. FALCON consists of two main parts: a warm-up phase and adaptive control in epochs phase. In the warm-up phase, using only a small amount of flow data (35 seconds-equivalent to approximately 85 vortex shedding interactions) FALCON recovers a succinct Fourier basis that explains the collected data and enforces that this learned basis is mostly composed of low-frequency components following the prior observations on turbulent flow dynamics. It then uses this basis to learn the unknown linear coefficients that best fit the acquired data on the learned Fourier basis during the adaptive control phase.

In the control design, FALCON uses model predictive control (MPC) and efficiently solves a short-horizon planning problem at every time step with the learned system dynamics. This recurrent short-horizon planning approach allows FALCON to adapt to the sudden changes in the flow while designing more sophisticated policies that consider future flow effects in contrast to the conventional purely reactive controllers. Moreover, the simple yet physically accurate dynamics learning approach of FALCON further facilitates the effective control design, which results in a sample-efficient and high-frequency control policy. During the adaptive control phase, FALCON refines its model estimate, i.e., the linear coefficients, in epochs in order to improve the learned model, which in turn improves the performance of the MPC policy. Overall, FALCON provides a simple, efficient, and interpretable dynamics modeling and an adaptive policy design method for the flow-predictive aerodynamic control problem and significantly outperforms the state-of-the-art model-free RL methods and conventional control strategies, i.e., PID, using only a total of 9 min of training data representative of approximately 1300 vortex shedding cycles. FALCON easily incorporates the physical and safety constraints in the policy design and builds on a fundamental understanding of how well nonlinear systems can be approximated and how these approximation errors affect the control performance, which we support with rigorous theory.

We implement FALCON on an experimental aerodynamic testbed that abstracts the fundamental physics involved in flight through a turbulent atmospheric flow and is specifically relevant for fixed-wing UAV applications. This testbed consists of a 3D-printed airfoil with actuated trailing edge flaps and an array of pressure sensors to measure the surrounding flow (Fig. 1B). The system is mounted in a closed-loop wind tunnel on a load cell measuring the aerodynamic lifting force acting on the airfoil. The testbed is placed in the wake of a bluff body at a Reynolds number of 230, 000, which generates a highly turbulent and vortical environment (Fig. 1C). This setting is on the upper-intermediate range of turbulent spectrum, and follows the description of realistic fixed-wing UAV flights with indicated Reynolds numbers ranging from 30, 000 to 500, 000[41]. The aerodynamic control goal is set to minimize the standard deviation of the lift forces by adjusting the position of the trailing-edge flaps in response to incoming disturbances with the help of flow sensors (Fig. 1E). In free flight, this would be equivalent to minimizing the inertial deviations along the lifting axis.

Through these wind tunnel experiments, we report that FALCON achieves 37% better disturbance rejection performance than the state-of-the-art model-free RL method[35], using only a single trajectory and 8 times less data. Moreover, we document a performance improvement of 45% over the conventional reactive PID controller. Overall, we find that the superior performance of FALCON is consistent over independent runs in the highly irregular unsteady turbulent flow dynamics, demonstrating the adaptation and generalization capability of FALCON to the unseen conditions.

Results

Novel Model-based RL Framework: Fourier Adaptive Learning and Control (FALCON)

FALCON has two main phases: a warm-up and an adaptive control in epochs phase (Fig. 2). The warm-up phase is a short initial period, where FALCON collects initial data about the fully unknown system. The goal is to purely explore the system and recover a coarse model of the dynamics. To that end, FALCON executes smooth and safe actions, i.e., time-correlated Gaussian inputs, that safely excite the system (see Methods for other variants). Since FALCON relies on pressure sensors on the airfoil to measure the system dynamics, it operates under partial observability. The chaotic dynamics of the systems undergo quick transients from their earlier states, forgetting them, making the current dynamics more heavily dictated by recent observations and actions than earlier ones. To overcome the uncertainties that partial observability brings, FALCON uses the recent history of actions and measurements to model the system dynamics. At the end of the warm-up phase, FALCON uses the data collected to learn the most relevant Fourier basis that explains the observed turbulent dynamics. FALCON is data efficient and requires only 35 s of flow data during the warm-up phase to recover an informative Fourier basis. This 35 s of flow data include fewer than 1500 samples taken over a period spanning approximately 85 vortex shedding events.

FALCON incorporates several key features that achieve data and computational efficiency in the basis learning process. In particular, FALCON uses ℓ_1-constrained (sparse) Fourier basis, as well as least absolute shrinkage and selection operator (lasso)[42] to recover a succinct basis representation (see details in Methods). This improved basis selection yields a significantly compact model representation while allowing physically accurate modeling of the underlying system dynamics due to the low-frequency dominant choice of Fourier basis, i.e., sparse Fourier basis vectors[43]. Indeed, spectral methods and modal analyses for modeling turbulent fluid dynamics are well-established concepts[44,45], and it is known that large eddies with low frequencies contain the most energy in turbulent flows[40]. This inductive bias in modeling via sparse Fourier basis reduces the number of samples required to learn the turbulent dynamics with small modeling errors and alleviates the computational burden in the predictive control design, facilitating high-frequency control actions. FALCON allows flexibility in the basis learning procedure such that the number of Fourier basis used in model learning could be easily adjusted based on the prior knowledge

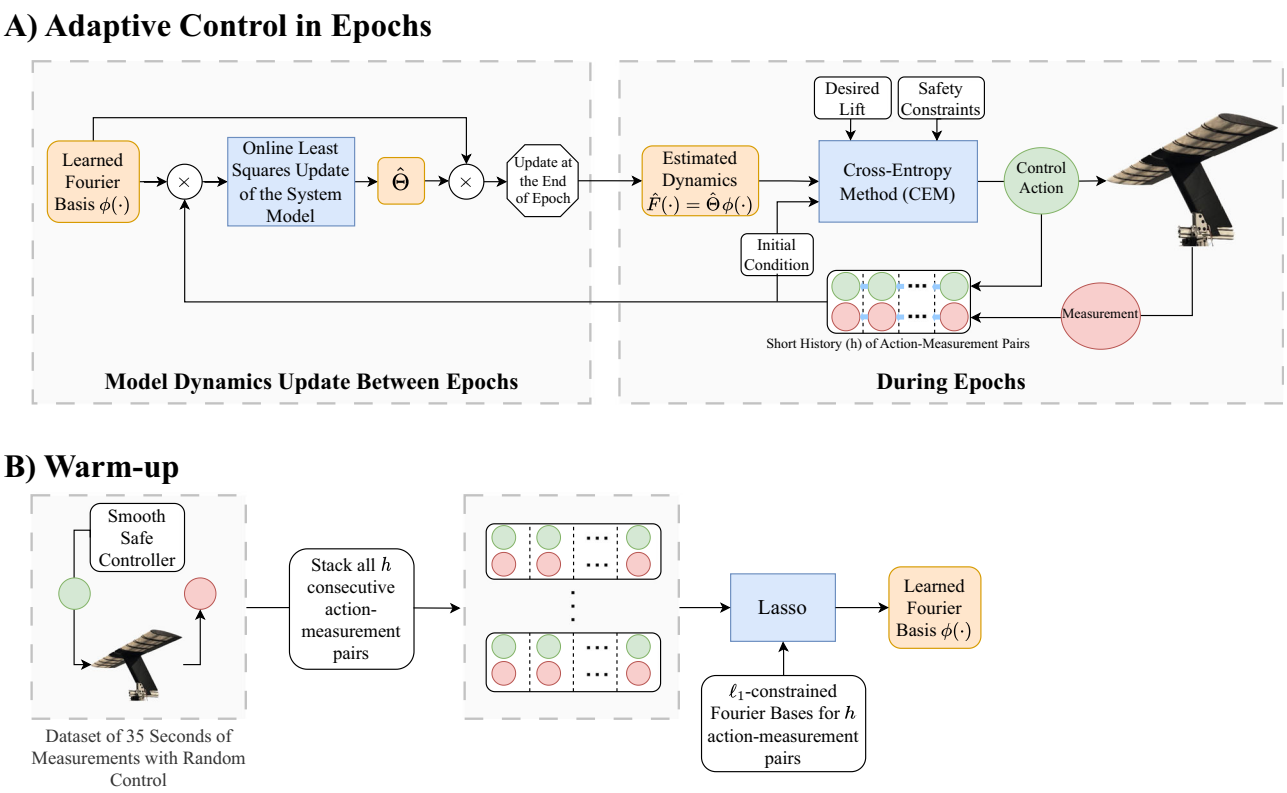

Fig. 2 | FALCON Framework. It consists of two phases: Warm-Up and Adaptive Control in Epochs. **A** Adaptive Control in Epochs: FALCON models the system dynamics as a linear map of the representation of a short history (h-length) of action-measurement pairs in the succinct Fourier basis learned in the warm-up phase. FALCON learns the unknown linear coefficients that best model the dynamics via online least squares. It updates the estimated system dynamics, i.e., the linear coefficients, at the end of each epoch, and during the epochs, it uses Cross-Entropy Method (CEM), a sampling-based MPC method, to control the airfoil under extreme turbulence using the estimated system dynamics while satisfying desired lift and safety requirements. **B** Warm-up: It is a one-time 35 s process before starting the adaptive control phase for safely collecting some exploratory data about the unknown system to recover a relevant Fourier basis to be used in learning and adaptive control. To achieve this FALCON forms h-length subsequences of action-measurement pairs (a short history) from the safely collected dataset and solves the lasso problem on the ℓ_1-constrained Fourier basis representation of these sub-sequences. FALCON selects the Fourier basis vectors that correspond to non-zero coefficients in the solution of the lasso problem as the succinct Fourier basis $\phi(\cdot)$ for the entire adaptive control in epochs phase for learning and control of the system.

of the system dynamics, the difficulty of the learning task, and the computational budget.

After recovering a succinct Fourier basis for model learning, FALCON starts the adaptive control in epochs phase, Fig. 2A. It estimates the model dynamics as a linear model in the learned Fourier basis and aims to learn the unknown linear coefficients that best fit the acquired data onto this basis. In particular, FALCON solves an online least-squares problem that has a closed-form solution to learn these linear coefficients. This interpretable and lightweight model learning allows online and/or batch updates for computational efficiency and comes with strong learning theoretical guarantees for the robustness of modeling (see Materials).

During this phase, FALCON designs an online control policy based on this learned model while improving the system dynamics model in an online fashion over time. This process goes in epochs with doubling duration, i.e., each epoch is double the length in seconds of the previous epoch, where at the end of each epoch FALCON updates its linear coefficient estimates on the model for better-refined dynamics modeling and control. This epoch schedule reduces the number of model updates towards the later stages of adaptive control where the dynamics are already well-modeled and only small tuning is required to further improve. We would like to highlight that FALCON is a single trajectory algorithm in the sense that it does not require a reset between epochs, which makes it efficient in the data collection process.

As the online control policy, FALCON uses model predictive control (MPC) with the estimated model dynamics to design the control inputs during the adaptive control phase. For controlling nonlinear dynamical systems such as aerodynamic control in turbulent flow considered in this work, finding the optimal solution to the control problem is usually challenging[46]. As a practical and efficient alternative, MPC policies have been the dominant choice for designing controllers in nonlinear dynamical systems[47]. Given the initial conditions, the transition dynamics (can be an estimated model or a nominal model), the running costs, and the terminal costs at any given time step, the objective in MPC is to solve a short horizon optimal control problem and execute the first action of the solution sequence. This process is then continued as we gather new observations. Intuitively, instead of trying to solve the challenging global optimal control problem, MPC myopically solves a locally optimal control problem. Usually physical or safety constraints on actions, observations, and dynamics are added in the MPC formulation due to its simplicity of implementation. The choice of the MPC policy depends on the control task. In general, the MPC policies are either optimization-based[48] or sampling-based[49]. However, sampling-based methods are usually preferred in model-based RL due to challenging nonlinear system dynamics and complicated cost and constraint functions[50].

Therefore, at every time step of the adaptive control phase, FALCON deploys a Cross-Entropy Method (CEM) policy, a sampling-based MPC policy[49], to design control actions using the most recent system dynamics estimate as the transition dynamics. CEM maintains a distribution, predominantly Gaussian, to sample action roll-outs for the short planning horizon and iteratively updates this distribution to assign a higher probability near lower cost action sequences based on the estimated system dynamics. After a certain number of updates, it executes the first action on the lowest cost-achieving action sequence in the sampled roll-outs (see further details of MPC design and CEM in particular in Methods).

FALCON takes in the most recent short history of actions and measurements as the initial condition for short-horizon MPC objective. For the running and terminal control costs FALCON can utilize any kind of cost functions as long as they can be evaluated efficiently depending on the control task. In our experiments, we design the cost function of FALCON based on our aerodynamic control objective such that FALCON avoids large lift forces and prevents rapid changes in lift forces and fast/jittery action changes. The first two design choices are clear from the control goal, i.e., minimizing the mean and the standard deviation of the overall lift forces, whereas, the last one is more subtle. In our experiments, we observed that non-smooth changes in actions cause additional lift forces on the airfoil (see further discussion on cost design choice in Materials). Furthermore, in the policy design FALCON includes action constraints due to mechanical restraints of the aerodynamics testbed as we shortly discuss in the next section.

FALCON can easily include further safety or physical constraints within its MPC framework. This makes FALCON a reliable algorithm for safety-critical tasks such as free flight through turbulence. Moreover, the recurrent short-horizon planning approach through CEM allows FALCON to design sophisticated policies that consider future flow effects through the use of estimated model dynamics. Thus, rather than designing purely reactive policies that cancel out the instantaneous aerodynamic forces, FALCON designs flow-predictive disturbance rejection policies which aim to minimize the lift forces while accounting for unsteady flow dynamics. In this way, FALCON adapts to sudden changes in the flow while avoiding overcompensation of the aerodynamic disturbances by maintaining an overall understanding of the flow field. The simple, yet physically sound and accurate dynamics learning approach of FALCON facilitates this effective control design, which results in a sample-efficient and high-frequency (42 Hz) state-of-the-art control policy with generalizable performance.

The construction of FALCON is modular such that different basis functions, e.g. wavelets[51], could be utilized in learning the underlying system dynamics depending on the domain knowledge about the system, while the MPC framework could be selected based on the specific needs, e.g. optimization-based MPC for simpler model dynamics. This interchangeable design of FALCON makes it a viable model-based RL method for designing diverse online/adaptive control strategies for various tasks (see Discussion). Moreover, it also allows for the derivation of strong theoretical guarantees for the robustness of model learning and the control performance under modeling error, see the Methods section. In particular, we prove that a wide range of partially observable nonlinear dynamical systems such as dynamical systems governed by partial differential equations could be learned with arbitrary modeling error using Fourier basis for FALCON. We also show that this effective model learning allows stable control design for robust MPC frameworks and the systems controlled by FALCON follow a trajectory close to the systems regulated with the same MPC policy that has access to *perfect* system dynamics information. Finally, we formalize the performance guarantee of FALCON such that the control performance of FALCON converges to the idealized MPC controller that knows the perfect system dynamics. These rigorous theoretical results display the reliability of FALCON while attaining state-of-the-art performance in predictive flow control.

Experimental aerodynamic testbed

In this work, we abstract the problem of stabilizing an aerodynamic system under turbulence to basic components while maintaining the core complexity of the physics involved. We utilize an experimental aerodynamics testbed that captures and generalizes the fundamental physics involved in flight through a turbulent environment[35]. The aerodynamic testbed consists of a symmetric generic airfoil with motorized trailing-edge flaps and integrated flow sensors (Fig. 1B). The trailing-edge flaps have an actuation range of [-40°, 40°], and are mapped linearly from the action space of $[-1, 1]$, yielding 1-dimensional control action per time step. Similar to flap systems on conventional airplanes, actuating the trailing-edge flaps generates a lifting force that can offset the aerodynamic forces associated with flow disturbances as shown in Fig. 1E. The testbed is equipped with nine sensors which are placed 10cm apart along the spanwise axis. Observations of the surrounding flow are measured through a series of eight pressure tap flow sensors built into the body of the airfoil, with a single pitot-static tube located at the center of the airfoil. The pressure taps, placed near the leading edge of the airfoil, provide valuable information on incoming pressure differentials between the upper and lower surface of the airfoil. The pitot-static tube measures the total pressure of the incoming flow which is approximately proportional to the mean velocity of the flow. The aerodynamic testbed is mounted on a one-dimensional load cell which is used to observe the lift forces acting on the airfoil, which serves the same effective role as an inertial measurement unit on conventional UAVs. Combined with nine flow sensors, we obtain 10-dimensional measurements per time step. Further details on aerodynamics testbed design are provided in Methods.

Turbulent environment

We study control in the context of a canonical problem in fluid dynamics: the turbulent wake of a bluff body. When placed incident to winds, bluff bodies produce an oscillating vortical wake commonly known as a Kármán vortex street[52]. At sufficient wind speeds, this wake becomes highly turbulent and can result in significant forces[43], as visualized with smoke in Fig. 1D. This photo is captured at Caltech Center for Autonomous Systems and Technologies (CAST) fan-array wind tunnel with a standard cylinder at a lower wind speed than the experiments presented in this work. Figure 1D depicts the turbulent wake of the cylinder where the vortex shedding is irregular. This phenomenon is famously responsible for the 1940 collapse of the Tacoma Narrows Bridge[53].

All of the quantitative results in this work were obtained from the experiments conducted in Caltech's Lucas Adaptive Wall Wind Tunnel, a closed-loop wind tunnel. As discussed previously, our experiments were performed in the wake of a bluff body at a mean flow speed of 6.81 m/s, which corresponds to a Reynolds number of $Re_D = 230{,}000$ over the bluff body. The bluff body consisted of a cylinder with a diameter of 30 cm with a normal flat plate fixed asymmetrically that increased the effective diameter to 53 cm, as shown in Fig. 1C. This construction is used to encourage vortex dislocation which results in less regular vortex shedding events[54]. Further, the bluff body was mounted to the walls of the tunnel with elastic cords to allow for dynamic oscillations which may also encourage less regularity in shedding events (see Methods for details).

We used particle image velocimetry (PIV) to visualize a portion of the turbulent flow field in our experimental environment (see Methods for details). Figure 3 presents the PIV measurements of the vorticity field contextualized in the wind tunnel. The complex vorticity patterns clearly demonstrate the chaotic and unsteady turbulent flow dynamics, with strong three-dimensional effects likely present in our experimental setting. Moreover, through hot-wire anemometer measurements in the wind tunnel, we record a turbulence intensity of 10.8%.

Baseline control methods

To test the performance of FALCON, we deploy several RL baselines and the industry-standard responsive control strategy of PID (Proportional-Integral-Derivative) control in our aerodynamics testbed. In particular, we compare FALCON with the twin delayed deep deterministic policy gradient algorithm TD3[55], its variant known as LSTM-TD3[56], and soft actor-critic algorithm SAC[57] (see Methods for a detailed overview of these methods). These methods are the state-of-the-art off-the-shelf model-free RL methods deployed in many real-world control tasks[34,35,58,59]. They are off-policy actor-critic algorithms that utilize neural networks for control policies. Off-policy methods are usually preferred over on-policy methods in real-world dynamical systems with unsteady dynamics since they can learn from a wide range of experiences, including observations from previous policies, which makes them more robust to changes in the environment. Another advantage of off-policy methods is that they can learn an optimal policy even when the current policy is significantly sub-optimal, which is usually the case for challenging real-world control tasks due to the lack of clearly superior expert

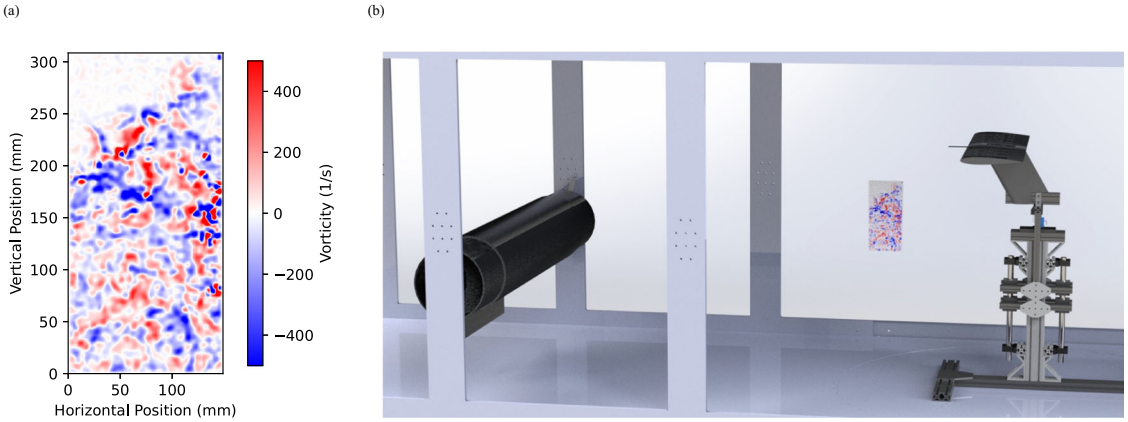

Fig. 3 | Turbulent Flow Field and Vorticity Measurements Inside the Wind Tunnel. a Particle Image Velocimetry (PIV) Visualization of the Turbulent Flow Field. **b** Depiction of PIV measurement location in the wind tunnel

policy. These all combined allow off-policy methods to be more stable during the learning process, which leads to better convergence and generalization performance[31].

The TD3 algorithm has previously demonstrated success in experimental flow control in different settings[34]. To improve performance in partially observable systems, such as the turbulent flow dynamics measured by sensors, LSTM-TD3 utilizes recurrent long-short-term memory (LSTM) cells in the neural network structure of TD3. The addition of LSTM cells has been previously shown to improve the performance in prediction and control of highly unsteady stochastic environments like turbulent flow fields[60]. In particular, recently, LSTM-TD3 has been demonstrated to achieve state-of-the-art performance in disturbance rejection in a similar experimental setting studied in this work[35]. Therefore, LSTM-TD3 provides the ultimate baseline for FALCON. In their implementations, both TD3 and LSTM-TD3 have nearly identical parameters besides the additional LSTM structure of LSTM-TD3 for an additional memory element in the policy. While TD3 and LSTM-TD3 provide deterministic policies, SAC designs stochastic policies, which is shown to achieve significant success in various real-world tasks such as quadrupedal robots and voltage control[57,61]. It provides a sample efficient alternative policy design method compared to TD3 and LSTM-TD3.

Due to the stochasticity in the process of training RL algorithms, we trained each of the agents presented here with three independent random seeds and present the average training results to display their performances. Unlike FALCON, the model-free methods work in episodes with reset for retraining. We train the model-free methods for 200 episodes of 800 samples per episode and use the best-performing agent for each algorithm in presenting their final performance. The feedback gains of the PID controller were tuned manually to achieve constant zero lift using the readings of load cell measurements. In our experiments, we run an exhaustive grid search over the feedback coefficients and report the best-performing controller. All methods, including FALCON, are implemented with 42 Hz sensing and control frequency.

Superior performance and sample efficiency of FALCON

First, we provide the results on the training of the methods. In presenting the training behavior, we exclude the warm-up of each method. Note that the warm-up phase of FALCON requires only 1500 samples, which corresponds to approximately 85 vortex shedding cycles from the upstream bluff body. Figure 4 shows the moving average of the mean and standard deviation of the lift forces on the airfoil for the best-performing policy of each method over the first 42,000 samples collected. In this plot, while the data collection procedure of FALCON does not pause in between model updates (epochs), the model-free algorithms pause (end of their episode) and train for some time to update their policy. From these plots, we observe that FALCON quickly finds the unknown linear coefficients to represent the system dynamics in the learned Fourier basis and achieves significantly better performance than model-free methods with fewer samples. We also note that during the 40,000 sample period shown, FALCON has only 25 learning updates while the other algorithms shown have 50. As the model-free algorithms used for comparison typically require more data, we trained these algorithms for a total of 200 episodes (equivalent to 160,000 samples), the remainder of which can be seen in Fig. 5. The training behavior of FALCON indicates that FALCON agents consistently improve throughout training and require significantly less training time to outperform state-of-the-art model-free methods due to its physically accurate model learning procedure and efficient control design.

Even though FALCON can suffer from model uncertainty and execute sub-optimal actions at the beginning of training due to unsteady flow dynamics (see the outlier bump in the standard deviation of lift in Fig. 4 around $t = 6000$), FALCON effectively explores the state-space to improve the accuracy of the model and hence the performance of the controller to bring the standard deviation in the lift forces to desirable values. After 10,000 samples, i.e., 4 min of training of FALCON, the average standard deviation of lift forces on the aerodynamics testbed remains stable at a level significantly better than other tested methods. Similarly, the mean lift forces achieved via FALCON consistently outperform that of model-free methods.

Among the model-free methods, Fig. 4 shows that LSTM-TD3 is the second best-performing algorithm by outperforming TD3 slightly, while SAC fails to achieve acceptable performance. Note that LSTM-TD3 and TD3 share the same policy constructions except the LSTM part that adopts latent states in the policy. Combined with the superior performance of FALCON, this highlights the importance of a latent state representation in achieving desirable learning and control performance in partially observable real-world settings. Similar to FALCON, LSTM-TD3 achieves consistent performance after sufficient samples, yet it requires an order of magnitude more than FALCON. Despite our significant efforts in hyperparameter tuning, SAC agents failed to learn desirable policies that minimize the aerodynamic forces. Even the best-performing agent significantly underperformed compared to other model-free policies, which indicates that stochastic policies such as SAC might not be suitable for controlling unsteady dynamical systems.

From our experiments, we observe that FALCON is more robust to hyperparameter tuning and has a notably more stable training process (Fig. 4) compared to model-free methods. In particular, in our training process, tuning FALCON requires only a few trials on the history of modeling (number of past observation-action pairs used), the sparsity weight in lasso for recovering a succinct basis, and the planning horizon for CEM. On the other hand, model-free methods require extensive hyperparameter search to achieve some learning behavior. This extensive search is significantly time-consuming and laborious in real-world problems, *e.g.* the training process of model-free methods corresponds to an hour of training for each hyperparameter configuration in our setting. This process becomes

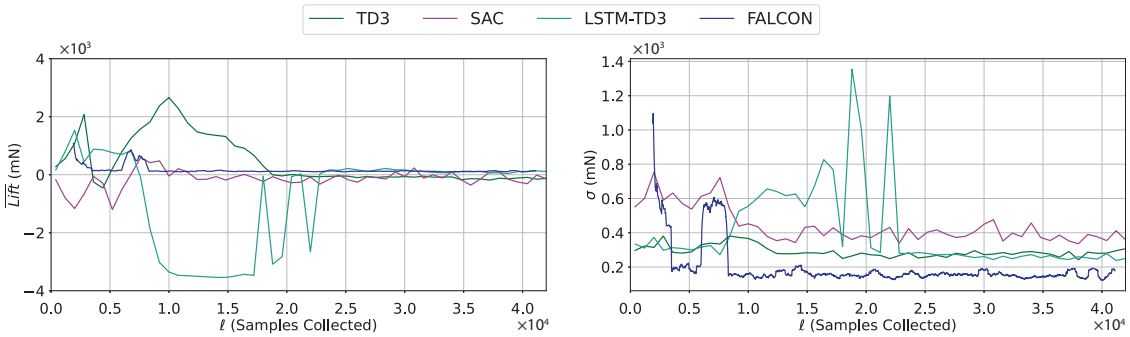

Fig. 4 | Evolution of the mean ($\overline{\text{Lift}}$) and standard deviation (σ) of the lift forces for the best-performing agents of each algorithm shown over the first 40,000 samples. The full training performance for the model-free algorithms can be found in Fig. 5.

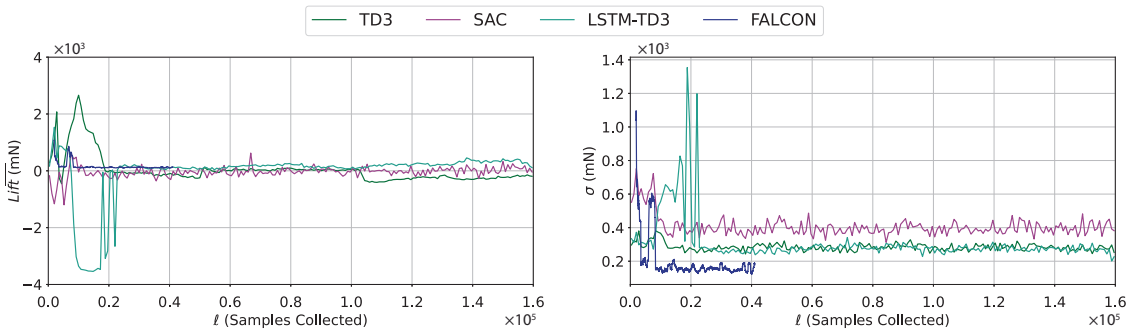

Fig. 5 | Evolution of the mean ($\overline{\text{Lift}}$) and standard deviation (σ) of the lift forces for the best-performing agents of each algorithm shown over the full 200 episodes for which the model-free algorithms were trained (160,000 samples).

unfeasible in online resource-constrained settings, which are typically the scenarios for adaptive control systems.

Our experiments overall showed that FALCON consistently achieves better and more stable training performance than model-free methods while converging to its optimal policy with orders of magnitude fewer samples. This superior performance and sample efficiency show that the simple yet efficient partially observable dynamics modeling approach of FALCON reduces the complexity of the aerodynamic control under turbulence problem significantly. Combined with the efficient predictive control design, FALCON agents learn to reject the flow disturbance effectively. The model learned by FALCON also aligns well with established knowledge regarding spectral energy content; in particular, the improved basis selection with ℓ_1-constraint and lasso enforces the model to have relatively low frequencies corresponding to the dominant energy-containing eddies. In our experiments, we observe that FALCON recovers model estimates which put significant weight on the low-frequency basis, e.g. DC components, and some high-frequency components (see Methods). This shows that via using the relevant basis for learning, FALCON learns a physically meaningful model, which contributes to training stability and disturbance rejection performance of FALCON.

To further investigate the effect of the concise Fourier basis-based model learning approach of FALCON, we implemented a reasonably sized deep neural network for model learning and combined it with CEM-based policy design as FALCON. However, despite our significant efforts in tuning, this approach failed to learn a reliable model for control design due to unsteady and chaotic flow conditions, which resulted in a performance significantly worse than the reported algorithms in this work. This outcome highlights that black-box dynamics modeling methods such as deep neural networks can fail in unsteady systems such as turbulent flow dynamics.

Consistent and generalizable performance of FALCON

Next, we study the generalization performance of FALCON and other methods including the PID controller. Table 1 presents the average performance of the best-performing policies by each method in 10 independent 90-second length runs, i.e., 4000 samples, as well as the number of samples required to train the respective "best" policies. We report the mean and standard deviation of the lift forces on the airfoil averaged over these runs. As discussed before, the standard deviation of the lift forces is the key metric for disturbance rejection and aerodynamic flow control. Table 1 shows that FALCON improves upon the prior state-of-the-art performance in flow disturbance rejection under extreme turbulence by 37%. Further, FALCON achieves this performance using only 8.7 min of data, whereas the model-free algorithms can take hours to train in the same setting[35]. This significant improvement with 8 times fewer samples shows that FALCON adapts and generalizes across independent runs despite remarkably different training conditions due to unsteady and chaotic turbulent flow dynamics. Moreover, FALCON outperforms the industry standard PID controller as well as TD3 policy by more than 45%. Similar to the training

Table 1 | Disturbance rejection performance of the methods over 10 independent 90 seconds test runs

		Average Over 10 Tests		Std Over 10 Tests	
Control Method	**Training Samples**	**Absolute Mean Lift (mN)**	**Std in Lift (mN)**	**Absolute Mean Lift (mN)**	**Std in Lift (mN)**
PID	*N/A*	6	271	1	8
TD3[55]	1.71×10^5	183	267	4	9
SAC[57]	1.51×10^5	268	395	213	64
LSTM-TD3[35]	1.76×10^5	139	236	16	5
FALCON	$\mathbf{2.20 \times 10^4}$	**2**	**148**	**10**	**13**

The best average absolute mean performance achieved by FALCON is highlighted in bold.

performance, we have observed that the SAC policy failed to achieve acceptable performance in disturbance rejection while requiring less training data to converge compared to other model-free methods. This result also suggests that stochastic policies might not be effective in controlling unknown unsteady or chaotic systems.

We also document that FALCON achieves the best average absolute mean performance. In particular, it outperforms PID control which is designed to keep the absolute mean close to zero. Even though it is a secondary and easy-to-offset metric in aerodynamic control, we observe that model-free methods attain significantly higher absolute mean lift compared to FALCON and PID controllers. Among these methods, LSTM-TD3 also achieves the lowest absolute mean.

Finally, we present the standard deviation in these performance metrics over 10 independent runs to test the consistency of the control methods' performance in Table 1. We see that the performance of FALCON is consistent over the unsteady dynamics with minimal change over the runs and it almost matches the consistency of the non-learning-based PID controller. Notably, besides SAC, other RL methods also perform consistently over these independent runs, where LSTM-TD3 which uses memory units in their policy construction, akin to FALCON, outperforms TD3. These results overall show that FALCON is able to generalize its performance to unseen disturbances and consistently provides the state-of-the-art predictive-flow disturbance rejection in extreme turbulent flow dynamics.

Discussion

We have designed and demonstrated FALCON, the first model-based RL method that can effectively learn to control aerodynamic forces acting on an airfoil under extreme turbulence with which conventional methods struggle. Our results indicate that combining flow sensing with physically sound model learning and efficient control design allows state-of-the-art disturbance rejection despite the chaotic nonlinear turbulent dynamics. Besides the superior performance, the physics-informed lightweight design for learning and control of FALCON allows an order of magnitude improvement over the number of samples required to achieve desirable control performance compared to prior RL methods. Further, we document that our method has a stable training procedure and a consistent performance even under highly irregular unsteady dynamics of turbulent flow. These results indicate the potential to use this method to stabilize systems such as UAVs under extreme turbulence in free-flight scenarios.

We observed that FALCON improves the aerodynamic control performance of prior state-of-the-art LSTM-TD3[35] by 37%. Even though LSTM-TD3 is a model-free RL algorithm, it shares a similarity with FALCON that it uses recurrent LSTM cells to utilize a history of observations in the policy design. With this construction, LSTM-TD3 is able to capture the latent state dynamics in designing policies. This allows LSTM-TD3 to handle partial observability in system dynamics due to sensor measurements, and design policies based on modeled latent states. Due to the structural similarities of LSTM-TD3 and TD3 algorithms, the superior performance of LSTM-TD3 over TD3 should be attributed to the latent state modeling via LSTM cells.

In contrast to FALCON, the modeling of latent states in LSTM-based policy design is mostly black box and without any physical interpretation. On the other hand, in FALCON, the flow dynamics are captured by significant low-frequency and some high-frequency model components with learned linear mixing coefficients using a history of observations and actions. Our proposed modeling approach in FALCON is motivated by the prior studies on turbulent flow dynamics that observe a well-defined frequency spectrum with significant low-frequency energy content for highly turbulent flows[35,40]. The interpretable dynamics modeling of flow disturbances of FALCON simplifies the disturbance rejection task to a low-dimensional learning-to-control problem, where learning and control design are executed efficiently. Moreover, this principled approach allows theoretical guarantees on sample-efficient model learning, robustness against imperfect learning, and control performance under modeling error which are derived for FALCON in the Methods section. All these results highlight the importance of deploying domain knowledge in model learning for unsteady and chaotic systems such as turbulent flow fields.

The industry-standard method to tackle similar problems relies on extensive engineering efforts in developing/tuning complex models that are often accompanied by non-physical domain jargon parameterization. These advanced and many decades-long efforts are behind many successes observed in practice on large aircraft, including Boeing 787 gust rejection. In our current work, we establish early learning methods for similar problems. In the future step of learning-based endeavors and studies, we plan to collaborate with industry partners to advance the learning methods and further address additional real-world challenges, enabling field experts to tackle daily design programs.

FALCON exhibits a modular structure, where the model learning and control components could be replaced depending on the task. The Fourier basis is deployed in FALCON and our experiments due to prior studies which showed that turbulent flow dynamics have a well-defined power spectrum dominated by the low-frequency components. Due to such underlying physics, the choice of Fourier basis allows theoretically guaranteed learning of the underlying system (see Methods). In particular, we rigorously show that the modeling error of such underlying systems could be made arbitrarily small with sufficient basis and data points from the system. This approach is in contrast to black-box modeling of the system dynamics, e.g. via deep neural networks, which naively uses purely data-driven basis functions, which may cause instability and fragility in model learning and control. In fact, our experiments with deep neural network modeling showed that in such temporally unsteady systems, it is hard to fully characterize the system dynamics without incorporating domain knowledge. This insufficient learning caused significantly inferior control policies.

The modeling capabilities of FALCON could be improved by adding nonlinearities to the modeling via Fourier basis. The composition of Fourier basis learning with nonlinear functions has shown success in learning the solution operators of partial differential equations[62]. Adopting such a modeling approach could further extend the model learning capabilities of FALCON and improve its aerodynamic control performance. Different basis vectors such as Random Fourier Features (RFF) have been deployed in prior model-based RL works[63]. Incorporating them along the Fourier basis can also extend the class of systems that could be learned via FALCON. Finally, different modeling approaches, such as modeling the pressure/lift differences on the sensors via the history of observations and actions, could be deployed to improve the sample efficiency and performance further. In our experiments, we tried this approach but did not observe a change. Yet, this approach could be helpful in deploying FALCON in more challenging turbulent environments.

In the control design, FALCON adopts CEM, a sampling-based MPC method, to exploit the learned accurate model. By design, CEM provides a transparent control design method in terms of what control cost is to be minimized and for how long of a trajectory should be considered in planning. This transparency is significant when incorporating domain knowledge and experimental observations directly in the control design. In particular, during our experiments, we observed that having rapid changes in the flap angles, i.e., too much variation in consecutive actions, results in a slight increase in lift forces on the airfoil. With this observation, we added a term in the cost design of FALCON which prevents these changes to a certain extent and improves aerodynamic control performance. This cost design is also intuitive for general flight control since it also reduces the wear and tear on the actuators. Moreover, due to this transparency, FALCON includes safety and physical constraints easily in the control design problem by simply eliminating trajectories or action sequences that violate these constraints.

This is in stark contrast with the black-box controllers provided by the model-free RL algorithms. These methods are very sensitive to many hyperparameters which control the neural network architecture and training procedure, yet, the effect of each hyperparameter on the performance is unclear. This lack of transparency leads to a reliance on intuition, experience, and trial and error when tuning these hyperparameters, making

the process time-consuming and frustrating. Even though the data presented for each model-free method took around 6 hours in the wind tunnel through training and testing, the actual process of hyperparameter tuning required dozens of additional hours for each algorithm. This presents a challenge in dynamic experimental environments, such as aerodynamic control under turbulence. On the other hand, the whole process of tuning, training, and testing of FALCON took around 9 wind tunnel hours in total.

In the aerodynamic control problem studied in this work, we considered 1-dimensional control actions with a 5 time-step planning horizon. This results in a relatively small search space to find optimal actions for CEM. This was particularly important in the control design of FALCON since the sampling-based MPC methods as CEM can be inefficient in longer planning horizons or larger action spaces. One can increase the number of samples per iteration in higher dimensional control problems, yet this might cause delays in control and poor performance. In order to deploy FALCON in higher dimensional control settings, utilizing a more efficient model predictive control method based on first-order optimization might be useful. This can be easily achieved with the same model learning module of FALCON and replacing CEM due to the modular structure of FALCON. Interior-Point Methods (IP) and Sequential Quadratic Programming (SQP) are two well-known algorithms for numerically solving these nonlinear optimization problems[64]. For high-dimensional problems, they can be further improved to exploit the sparsity in the control design and achieve desirable performance without sacrificing efficiency[65]. SQP methods are particularly good candidates in the control design since they use the result of the previous iteration to warm-start the next iteration of the control design similar to CEM.

In the warm-up phase, the proposed approach of solving the lasso problem over ℓ_1-constrained Fourier basis is able to learn a concise and effective basis for representing the system dynamics in a data-efficient way. This requires only 35 s of flow data at 42 Hz which is collected with time-correlated Gaussian inputs. This is equivalent to approximately 85 vortex-shedding cycles, however, due to the irregular shape and dynamic motion of the bluff body, it is likely that the wing-vortex interactions varied significantly during this period. Finding the solution to lasso takes about 7 min on a standard desktop computer, a CPU-based MacBook Pro, and this problem is solved only once at the end of the warm-up phase. It is opposed to the often-the-case deep learning approaches that have limited applicability to the onboard control chip devices. This effective succinct basis representation for the underlying dynamics allows FALCON to design control actions in less than 10 ms within the CEM model predictive control framework which yields 42 Hz sensing and control frequency. The fast adaptive control approach allows FALCON react to the changes in the flow field rapidly while still reasoning about how to mitigate upcoming turbulent disturbances on the system via the learned and updated model dynamics. To achieve this fast control design, FALCON leverages the parallel computing on GPU and samples a significant amount of initial action roll-out in CEM to overcome possible local minima in designing control actions. In this end-to-end control loop, serial communications between the controller and sensors, and actuators are the main bottleneck.

To further improve the disturbance rejection performance of FALCON, increasing the control frequency is one of the future developments to focus on. This could be achieved by reducing the code execution time and communication delays. Deploying a faster implementation of FALCON using C++ or utilizing a more computationally efficient MPC framework such as CEM-GD[66] which combines zeroth and first-order optimization methods could allow us to achieve sub-5 ms control design duration. Moreover, having a streamlined communication layer could also reduce the latency between the controller and sensors or actuators significantly.

In this work, we have developed a model-based reinforcement learning method, FALCON, on a generic aerodynamic testbed for flow-informed aerodynamic control under extreme turbulence. FALCON was tested on a single-dimensional aerodynamic control system, yet it can be extended and adapted to systems with higher degrees of freedom. In particular, we can consider other forces and moments in three dimensions, besides the vertical lift forces acting on the testbed. Our experiments are conducted under the Reynolds number of Re_D = 230,000. In order to ensure the generalizable performance of FALCON across a range of Reynolds numbers, i.e., different turbulence characteristics, and different geometries of airfoils, further investigation is required.

The findings of this work hold potential in the deployment of next-generation technologies, including but not limited to flow-sensing UAVs capable of stable flight in windy urban areas and flow-informed wind turbines with gust protection. In the fixed-wing UAVs, FALCON is a promising algorithm for the inner-loop attitude control for fixed-wing vehicles. This will allow drones to maintain stable flight in extreme conditions by reducing the impact of turbulent disturbances. We believe that model-based RL methods, and FALCON in particular, could be used for full-stack control and navigation using flow information and simulated environments. The testbed in this work emulates stabilizing a UAV at a constant altitude. Future work will consider using FALCON with a trajectory planner such that FALCON aims to maintain the desired location coming from the planner and interacts with the planner to achieve energy efficient and safe navigation, similar to the prior work in computational fluid dynamics[33]. To accomplish this will require overcoming challenges such as sim-to-real transfer and distribution shift in data, where the data efficiency and fast adaptation capabilities of FALCON would be critical. We suggest using indicators of changes in turbulent conditions in hierarchical planning to control the frequency of model updates within FALCON. Another strategy would be using meta-learning to make the model learning process adaptive in basis selection and model updates for different turbulent conditions with different basis representations.

Methods

Overview

We model the underlying turbulent dynamics as the following discrete-time nonlinear time-invariant system:

$$y_{t+1} = f(u_t, \ldots, u_{t-h+1}, y_t, \ldots, y_{t-h+1}) + e_t. \tag{1}$$

Here $y_t \in \mathbb{R}^{d_y}$ denotes the observation (measurements) from the system at time t, $u_t \in \mathbb{R}^{d_u}$ denotes the control input at time t, and e_t is some process noise. We consider the setting where the dynamics are governed by an unknown nonlinear function $f : \mathbb{R}^{h(d_u+d_y)} \to \mathbb{R}^{d_y}$ that maps past h observations and inputs to the next observation and with an additive noise e_t. These systems are denoted as Nonlinear Autoregressive Exogenous (NARX) systems and they are the central choices of dynamics modeling for nonlinear systems due to their input and output-dependent parametrization[67]. In particular, the model given in (1) is an order-h NARX system and is mainly used to capture partially observable nonlinear systems with fading memory[68,69].

Let $C_{0:T}(\cdot, \cdot)$ denote the sequence of cost functions on inputs and observations, which define the objective for the controlling agent. The stochastic optimal control problem is defined as,

$$\min_{u_0,\ldots,u_T} \mathbb{E}\left[\sum_{t=0}^{T} C_t(y_t, u_t)|y_0\right] \tag{2}$$

subject to dynamics given in (1) with initial condition of y_0 and where u_t is chosen causally. For nonlinear dynamical systems such as (1), finding the optimal solution to this problem is usually challenging[46]. As a practical and efficient alternative, model predictive control (MPC) has been adopted for designing controllers in nonlinear dynamical systems[47]. In MPC, at any time step t, given the initial conditions, the transition dynamics (can be an estimated model $\hat{f}$), running and terminal cost functions $C_{t:t+\tau}(\cdot, \cdot)$, the planner solves:

$$\begin{aligned} &\min_{u_t,\ldots,u_{t+\tau}} \sum_{s=t}^{t+\tau} C_s(y_s, u_s) \\ &\text{s.t.} \quad y_{t+1} = \hat{f}(u_t, \ldots, u_{t-h+1}, y_t, \ldots, y_{t-h+1}), \end{aligned} \tag{3}$$

a short τ-step optimal control problem, and executes the first action u_t and continues this process as it gathers new observations. Intuitively, instead of trying to solve the challenging global optimal control problem, MPC myopically solves a locally optimal control problem (3). Note that (3) presents an unconstrained MPC problem, and usually physical or safety constraints are added to the formulation. This makes MPC a viable approach for control design in model-based RL, thus, we will adopt it in our control design. Since we do not know the underlying dynamics f, we need to learn it from the data collected from the system. We achieve this by learning the underlying system on a Fourier basis.

A Fourier series is an expansion of a periodic function in terms of an infinite sum of complex exponentials, or sines and cosines. They are one of the most popular choices of the set of basis in representing periodic functions or periodic extensions of functions in a bounded domain due to their ability to approximate functions arbitrarily[70]. Consider the domain $\Omega = (0, 2\pi)^d$ in $\mathbb{R}^d$. Let $W_p^{m,2}(\Omega)$ denote the Sobolev space of order m for periodic functions. For a nonlinear function (or *its periodic extension*), $\bar{F}(\cdot) : \mathbb{R}^d \to \mathbb{R}$, that lives in $W_p^{m,2}(\Omega)$, one can write its Fourier series as

$$\bar{F}(x) = a_0 + \sum_{\boldsymbol{\omega}} \left[a_{\boldsymbol{\omega}} \cos\left(\boldsymbol{\omega}^\top x\right) + b_{\boldsymbol{\omega}} \sin\left(\boldsymbol{\omega}^\top x\right) \right] \tag{4}$$

where $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_d]$, $\omega_j \in \{1, 2, \ldots\}$, $1 \le j \le d$ and $a_{\boldsymbol{\omega}}$, $b_{\boldsymbol{\omega}}$ are Fourier series coefficients. Note that this representation can be on infinitely many bases. However, in approximating $\bar{F}(\cdot)$, one can choose only a finite number of basis among $\boldsymbol{\omega}$ and find the best approximation on this basis. To this end, the popular choice is to consider the nth order Fourier expansion and approximate $\bar{F}(x)$ in $\boldsymbol{\omega}$ where $\omega_j \in \{1, \ldots, n\}$. This corresponds to $D = 1 + 2n^d$ basis functions and results in a D-dimensional Fourier series feature representation:

$$\phi(x) = \left[1, \cos(\boldsymbol{\omega}_1^\top x), \sin(\boldsymbol{\omega}_1^\top x), \ldots, \cos(\boldsymbol{\omega}_{(D-1)/2}^\top x), \sin(\boldsymbol{\omega}_{(D-1)/2}^\top x)\right]^\top. \tag{5}$$

One can choose the truncated Fourier series representation to approximate $\bar{F}(x)$ such that for

$$\mathbf{w} = [a_0, a_{\boldsymbol{\omega}_1}, b_{\boldsymbol{\omega}_1}, \ldots, a_{\boldsymbol{\omega}_{(D-1)/2}}, b_{\boldsymbol{\omega}_{(D-1)/2}}]^\top,$$

the approximation is $\mathbf{w}^\top \phi(x)$. However, this does *not* correspond to the best approximation in this basis in L^p-norms for $1 \le p \le \infty$[70]. For the best L^p-norm approximation using (5), one needs to solve for the optimal coefficients $a_0, a^*_{\boldsymbol{\omega}_i}$ and $b^*_{\boldsymbol{\omega}_i}$ for $i \in \{1, \ldots, (D-1)/2\}$.

FALCON algorithm

In this section, we present the methodology and the algorithmic details of our proposed model-based RL method Fourier Adaptive Learning and Control (FALCON). FALCON learns the model dynamics in Fourier basis through interaction with the system and deploys MPC using the learned model for control design. The outline of FALCON is given in Alg. 1. FALCON has two phases: Warm-up and Adaptive Control in Epochs.

Algorithm 1. FALCON

Algorithm 1 FALCON

Input: T_w, h, t_{ep}, τ, D, $C_{0:T}$

—— WARM-UP ——

for $t = 1, 2, \ldots, T_w$ **do**

Deploy exploratory u_t and store $\mathcal{D}_0 = \{y_t, u_t\}_{t=0}^{T_w}$

Form $s_t = [y_{t-1:t-h}^\top, u_{t-1:t-h}^\top]^\top$ for all t using $\mathcal{D}_0$

Compute $\phi'(s_t)$ via (5) $\forall s_t$, D'-dimensional Fourier Series representation

Solve lasso (10) to learn succinct Fourier basis $\phi(\cdot)$ representation of system dynamics

—— ADAPTIVE CONTROL IN EPOCHS ——

for $i = 1, \ldots$ **do**

Solve for $\hat{\Theta}_i$ via (7) & Form $\hat{F}_i(\cdot) = \hat{\Theta}_i^\top \phi(\cdot) \to$ Model Dynamics Update via Online Least Squares

for $t = T_w + (i-1)t_{ep} + 1, \ldots, T_w + it_{ep}$ **do**

$u_t = \text{MPC}(\hat{F}_i, y_{t:t-h+1}, u_{t-1:t-h+1}, C_{t:(t+\tau)})$

Observe y_{t+1} & Form s_{t+1} and $\phi(s_{t+1})$ using the Learned Fourier Basis

FALCON starts with a short warm-up period to collect some data about the unknown system. In this phase, the goal is to purely explore the system and recover a coarse model of the dynamics. Therefore, FALCON focuses on safely exciting the system for T_w time steps. The predominant choice for such a task is to use isotropic Gaussian inputs, $u_t \sim \mathbf{N}(0, \sigma_u I)$. However, for certain tasks, one may require smoother or safer exploration. This is usually the case in safety-critical control tasks like flight control under turbulence[35] or bipedal/quadrupedal walking[71]. In these situations, FALCON can use time-correlated inputs for smooth actions such that it avoids jerky and sudden changes in the actions. To this end, for some $\gamma \in [0, 1]$, FALCON can use the following control inputs

$$\begin{aligned} u_1 &= \boldsymbol{\eta}_1 \\ u_t &= \gamma u_{t-1} + \sqrt{1-\gamma^2}\boldsymbol{\eta}_t \end{aligned}$$

where $\boldsymbol{\eta}_t \sim \mathbf{N}(\mathbf{0}, \sigma_\eta \mathbf{I})$. We deploy this controller with $\gamma = 0.8$ during the warm-up phase in our experiments. Moreover, FALCON can deploy known safe nominal controllers, such as trajectory generators[72] or PID controller, accompanied with isotropic excitements, i.e., $u_t = K(y_t) + \boldsymbol{\eta}_t$ where $K(\cdot)$ is the nominal controller and $\boldsymbol{\eta}_t \sim \mathbf{N}(0, \sigma_\eta I)$.

After the warm-up, FALCON starts adaptive control of the underlying system. It uses epochs of doubling length starting with an initial epoch of t_{ep} time steps, i.e., each ith epoch is $2^{i-1}t_{ep}$ time steps for $i = 1, 2, \ldots$. FALCON is a single trajectory algorithm and does not require a reset between epochs. This makes FALCON efficient in data collection in the experiments.

At the end of the warm-up, FALCON estimates the model dynamics as a linear model in Fourier basis. To this end, it generates $T_w - h +$

1 subsequences of h input-output pairs,

$$s_i = \left[y_{i-1}^\top, \ldots, y_{i-h}^\top, u_{i-1}^\top, \ldots, u_{i-h}^\top\right]^\top \in \mathbb{R}^{h(d_y+d_u)}$$

for $h \le i \le T_w$. Using (1), one can write the system dynamics as $y_t = F(s_t) + e_t$. To estimate the unknown nonlinear function F, FALCON considers the nth order Fourier expansion of F and generates D-dimensional Fourier series representations of all s_t as given in (6), $\phi(s_t)$. The order of the Fourier expansion, thus the dimension D, is an important hyperparameter of FALCON. This choice depends on many factors including prior knowledge of the system dynamics, the difficulty of the learning task, and the computational budget. As explained in the Overview section of Methods, a wide range of nonlinear systems can be represented as linear models in the Fourier basis. Therefore, FALCON considers the following model for estimating the system dynamics F,

$$y_t \approx \Theta_*^\top \phi(s_t) + e_t, \tag{6}$$

for an unknown $\Theta_* \in \mathbb{R}^{D \times d_y}$. To recover an estimate of Θ_* solves a least-squares problem,

$$\min_{\Theta} \lambda \parallel \Theta\parallel_F^2 + \parallel Y_{T_w} - \Theta^\top \Phi_{T_w}\parallel_F^2 \tag{7}$$

for some $\lambda > 0$, where $Y_t = [y_t, \ldots, y_h] \in \mathbb{R}^{d_y \times t-h+1}$, $\Phi_t = [\phi(s_t), \ldots, \phi(s_h)] \in \mathbb{R}^{D \times t-h+1}$. The solution to this problem is given as $\hat{\Theta}_1 = (\Phi_{T_w}\Phi_{T_w}^\top + \lambda I)^{-1}\Phi_{T_w}Y_{T_w}^\top$. Using $\hat{\Theta}_1$, FALCON estimates the system dynamics as $\hat{F}_1(s) = \hat{\Theta}_1^\top \phi(s)$. FALCON repeats this dynamics estimation process at the beginning of each epoch using all the data gathered so far. Note that for large D, computing the closed-form solution could be computationally demanding or cause numerical errors. Instead, the model estimates can be updated recursively throughout the epochs using online updates, which we utilize in our implementation for the experiments. In particular, FALCON stores only the current model estimate, i.e., the model estimate at time step t in epoch i: $\hat{\Theta}_{i,t}$, and the inverse design matrix (sample covariance matrix), i.e., $V_{t-1}^{-1} = (\Phi_t\Phi_t^\top + \lambda I)^{-1}$. Using these the model estimates can be updated recursively throughout the epochs using online or batch updates via

$$\hat{\Theta}_{i,t} = \hat{\Theta}_{i,t-1} + \frac{V_{t-1}^{-1}\phi(s_t)\left(y_t - \hat{\Theta}_{i,t-1}^\top \phi(s_t)\right)^\top}{1 + \phi(s_t)^\top V_{t-1}^{-1}\phi(s_t)}, \tag{8}$$

where V_{t-1}^{-1} is also updated recursively[73],

$$V_t^{-1} = V_{t-1}^{-1} - \frac{V_{t-1}^{-1}\phi(s_t)\phi(s_t)^\top V_{t-1}^{-1}}{1 + \phi(s_t)^\top V_{t-1}^{-1}\phi(s_t)}.$$

Note that FALCON uses the initial most recent model estimate at the beginning of the epoch for the control design during the entire epoch. These online update rules are used to efficiently update the model estimates in the background at each time step with the new data such that at the beginning of the next epoch, there is no delay in updating the model estimate. This feature is important in real-time control systems where any delay in the system can cause further problems and compromise safety.

Note that as the order of the Fourier basis increases, D increases exponentially in the system's dimension. For large h, i.e., higher order NARX models, this may cause an additional computational burden. To remedy this, we propose to use ℓ_1-constrained Fourier basis and Least Absolute Shrinkage and Selection Operator, i.e., lasso[42], for an improved basis selection in FALCON after the warm-up period. In particular, instead of generating the bases $\boldsymbol{\omega_i}$s for all n, we only consider the ℓ_1-constrained bases, i.e., $\|\boldsymbol{\omega_i}\|_1 \le n$. The ℓ_1 constraint reduces the number of basis vectors from $1 + 2n^{h(d_y+d_u)}$ to $2D'$ basis vectors where

$$D' = \binom{h(d_y + d_u) + n}{n}. \tag{9}$$

We then solve the lasso problem for the warm-up samples that are represented in these ℓ_1-constrained Fourier basis vectors. Lasso is the ℓ_1-regularized least squares method to recover sparse models, with few non-zero coefficients. Given the data points gathered during the warm-up period T_w, using D' number basis vectors with $\|\boldsymbol{\omega_i}\|_1 \le n$, FALCON forms the following feature representations for $s_h, \ldots, s_{T_w}$ generated via T_w:

$$\phi(s_i) = \left[\cos\left(\boldsymbol{\omega_1}^\top s_i\right), \sin\left(\boldsymbol{\omega_1}^\top s_i\right), \ldots, \cos\left(\boldsymbol{\omega_{D'}}^\top s_i\right), \sin\left(\boldsymbol{\omega_{D'}}^\top s_i\right)\right]^\top.$$

FALCON then solves the lasso problem:

$$\min_W \frac{1}{2T_w} \parallel Y_{T_w} - W^\top \Phi_{T_w}\parallel_F^2 + \alpha \parallel W\parallel_1 \tag{10}$$

for some $\alpha > 0$. FALCON then uses the basis vectors that have nonzero feature coefficients (entries) in the solution of (10), $W_\star$, for learning the model dynamics in the adaptive control phase. The choice of α determines the sparsity of the model learned $W_\star$ which in turn determines the number of basis vectors, D, used in model learning, i.e., bigger α results fewer non-zero entries in $W_\star$ and fewer basis vectors for estimating the dynamics in the adaptive control period. This improved basis selection significantly decreases D and reduces the computational burden and the samples required to learn the dynamics.

Once FALCON has an estimated model, it uses an MPC policy to design the control inputs during the epoch. The choice of the MPC policy depends on the control task. In general, the MPC policies are either optimization-based[48] or sampling-based[49]. However, sampling-based methods are usually preferred in model-based RL due to challenging nonlinear system dynamics and complicated cost functions[50]. Thus, FALCON uses Cross-Entropy Method (CEM) as the MPC policy. CEM is a sampling-based (zeroth order) MPC policy to solve the problem given in (3). CEM maintains a distribution, predominantly Gaussian, to sample action roll-outs for the planning horizon and iteratively updates this distribution to assign a higher probability near lower-cost action sequences based on the estimated dynamics. After a certain number of updates (once it converges), it executes the first action on the lowest cost-achieving action sequence in the sampled roll-outs. The CEM algorithm is given in full detail in Algorithm 2.

Algorithm 2. Cross Entropy Method (CEM)

1: **Input:** $\tau, K, M, 0<\gamma<1, N, \sigma_{init}, \hat{F}(\cdot), y_{t:t-h+1}, u_{t-1:t-h+1}, C_{t:(t+\tau-1)}, \mathbf{N}(\mu, \sigma^2 I)$
2: **for** $i = 1, 2, \ldots, M$ **do**
3: **if** $i = 1$ **then**
4: Set the mean μ to the best action sequence from the previous time-step by shifting (Warm-Start)
5: Set the variance $\sigma = \sigma_{init}$
6: Sample $K\gamma^{i-1}$ action sequences $u_{t:t+\tau-1}^j$ of τ length using $\mathbf{N}(\mu, \sigma^2 I), j \in \{1, \ldots, K\gamma^{i-1}\}$
7: Compute the trajectory roll-outs $\forall u_{t:t+\tau-1}^j$ using $\hat{F}(\cdot)$ with initial $y_{t:t-h+1}$, $u_{t-1:t-h+1}$
8: Compute the cost of each trajectory roll-out using $C_{t:(t+\tau-1)}$
9: Sample best N action sequences according to their acquired costs
10: Update μ and σ to fit the Gaussian distribution to the best N action sequences
11: Execute the first action of (i) the best action sequence of the Mth iteration or (ii) a newly sampled action sequence using $\mathbf{N}(\mu, \sigma^2 I)$

At any time step t, FALCON uses the most recent dynamics estimate $\hat{F}_k(\cdot)$, the last h input-output pairs as the initial condition, and the next τ cost functions $C_{t:(t+\tau)}$ in solving the problem in (3) for the planning horizon τ. FALCON executes the first action u_t in the solution of (3), receives the

output y_{t+1}, and constructs s_{t+1} and $\phi(s_{t+1})$. FALCON repeats this adaptive control process throughout the epoch. Note that any safety or physical constraint can be easily included in the MPC policy design problem (3), which makes FALCON a reliable algorithm for safety-critical environments.

Implementation details of FALCON

We use an order-4 NARX model for learning the underlying system dynamics, $h = 4$. In our experiments, we deduce that this is optimal to overcome the uncertainties of partial observability and reasonable computational complexity. With this choice, s_t in the system modeling becomes 44-dimensional vector. To estimate the unknown nonlinear system F, we consider the 3rd order Fourier expansion. However, to reduce the computational complexity for such a high-dimensional learning problem, we only use $\|\boldsymbol{\omega}_i\|_1 \leq 3$ constrained basis vectors and use lasso to identify the most relevant basis vectors using the warm-up data as described in Appendix. At the end of this procedure, we obtain $D = 319$ dimensional Fourier series representation for learning the model dynamics.

The control goal in disturbance rejection is to minimize the mean and the standard deviation of the lift forces acting on the airfoil. Thus, we design our cost function to penalize large lift forces, rapid changes in lift forces, and fast/jittery action changes. FALCON has a warm-up duration of around 35 s, i.e., $T_w = 1500$ samples, using the time-correlated sum of Gaussian inputs for smooth exploration to collect some data about the unknown system dynamics and recover the most relevant Fourier basis. The epochs of the adaptive control period are approximately 38 seconds, $t_{ep} = 1600$ samples per epoch. FALCON uses Cross-Entropy Method (CEM) as the MPC policy. CEM is a sampling-based MPC policy to solve the problem given in (3)[49]. CEM maintains a distribution, predominantly Gaussian, to sample action roll-outs for the planning horizon and iteratively updates this distribution to assign a higher probability near lower-cost action sequences based on the estimated dynamics. After a certain number of updates, it executes the first action on the lowest cost-achieving action sequence in the sampled roll-outs. The CEM algorithm is given in full detail in Algorithm 2.

We compare FALCON with several model-free RL methods, including TD3[55], LSTM-TD3[56], SAC[57], and the industry-standard responsive control strategy of PID (Proportional-Integral-Derivative) controller. Of all these algorithms, LSTM-TD3 has been demonstrated to achieve state-of-the-art performance in disturbance rejection[35]. Unlike FALCON, the model-free methods work in episodes with reset for retraining. We train the model-free methods for 200 episodes of 800 samples per episode and test the best-performing policy in presenting the results. All methods, including FALCON, are implemented with 42 Hz sensing and control frequency.

Adaptive control design

FALCON uses CEM as the MPC policy. CEM is a sampling-based (zeroth order) MPC policy to solve the problem given in (3)[49]. CEM maintains a distribution, predominantly Gaussian, to sample action roll-outs for the planning horizon and iteratively updates this distribution to assign a higher probability near lower-cost action sequences based on the estimated dynamics. After a certain number of updates (once it converges), it executes the first action on the lowest cost-achieving action sequence in the sampled roll-outs. For the planning horizon, FALCON uses $\tau = 5$ in CEM. Furthermore, FALCON samples $K = 1000$ trajectories in the first action roll-out of CEM and decays the number of samples in each update. In prior works, this sampling strategy has been observed as an efficient way to avoid possible local minima in finding the optimal action actions[66]. The CEM algorithm is given in full detail in Algorithm 2.

Table 2 summarizes the hyperparameters for FALCON in our experiments. In order to achieve the desired control and sensing frequency number of CEM samples (K) and iterations (M) create a trade-off in the implementation. Maintaining this control and sensing frequency is crucial in order to avoid *undersampling* the turbulent dynamics.

Table 2 | Hyperparameters of FALCON in our experiments

Hyperparameter	Range	Best
NARX-order (h)	3–5	4
Fourier Series Coef. (D)	100–700	319
Planning Horizon (τ)	3–8	5
CEM Samples (K)	150–1500	1000
CEM Iteration (M)	4–7	6
CEM Number of Elites (N)	10–30	30

Table 3 | Comparison of Works with Regret Guarantees in Nonlinear Dynamical Systems

Work	Regret Result	Learning Basis	Computational Efficiency	Memory Efficiency
85	$\sqrt{T}$	Known	No	No
63	$T^{2/3}$	Unknown	Yes	Yes
This Work	$\sqrt{T}$	Unknown	Yes	Yes

Stability and performance guarantees

1. We derive the learning guarantees for using the Fourier series as a basis for learning the dynamics. In particular, we prove that FALCON learns any partially observable nonlinear system that belongs to an extended Sobolev space of periodic functions with the near-optimal estimation error rate of $\tilde{O}(T^{\varepsilon-0.5})$, after T samples where ε depends on the smoothness of the Sobolev space and $0 \leq \varepsilon < 0.5$.
2. We show that FALCON attains $\tilde{O}(\sqrt{T})$ regret against the agent who has access to the underlying dynamics and uses the same control design. To the best of our knowledge, FALCON is the *first* efficient RL algorithm that achieves $\tilde{O}(\sqrt{T})$ regret in online control of nonlinear dynamical systems, Table 3.

Most of the model-based RL methods with guarantees are developed for linear systems due to their simplicity[74–82]. The central goal of these works is to derive finite-time learning and regret guarantees. Recently, there has been a growing interest to extend these results to nonlinear systems.[83,84] consider the model learning problem by modeling the underlying system as a linear function of a *known* nonlinear basis.[85] study the regret minimization in this setting and propose an approach which attains $\tilde{O}(\sqrt{T})$ regret, but is *not* computationally or memory efficient.[63] use kernel approximation to learn the underlying system and design an efficient RL algorithm, yet, achieve $\tilde{O}(T^{2/3})$ regret. Further, the empirical performances of these methods are demonstrated *only* on simulation. FALCON provides significant improvements upon these prior works in terms of regret guarantee and efficiency (Table 3). To the best of our knowledge, FALCON is the first efficient RL algorithm to attain $\tilde{O}(\sqrt{T})$ regret in partially observable nonlinear systems and achieve effective performance in a challenging real-world task.

In this section, we provide the learning and regret guarantees of FALCON. The technical details and the proofs are given in Supplementary Material. First, let $F_i(\cdot) : \mathbb{R}^{h(d_y+d_u)} \to \mathbb{R}$ denote the ith mapping of F from input to output, i.e., $y_{t,i} = F_i(s_t) + e_{t,i}$. We assume the following regularities on the system.

Assumption 1. The system $F(\cdot)$ is L-Lipschitz and (λ, α, K)-exponentially input-to-output stable (e-IOS), i.e.,

$$\| \, |\mathbb{E}[y_t | y_{t_0}, u_t, \ldots u_{t_0}]\| \leq \lambda \alpha^{t-t_0} \| y_{t_0} \| + K \sup_{i \in [t_0:t]} \| u_i \|,$$

for $t > t_0$, $\lambda, K > 0$ and $0 < \alpha < 1$. Moreover, F_i (or its periodic extension) lives in $W_p^{m,2}([0, 2\pi]^{h(d_y+d_u)})$, for all $1 \le i \le d_y$.

This assumption is inspired by the chaotic nature of the systems that indicate a fast transitory phase from their earlier states, making the current dynamics forget the much earlier states. Please note that if the system dynamics show complex behavior that does not satisfy this assumption, the method introduced in this work is no longer directly applicable. The first assumption is required to avoid blow-ups in the output due to noise and unmodeled dynamics. The Sobolev space assumption guarantees that the underlying system can be represented on the Fourier basis. For simplicity, we assume that $e_t \sim \mathbf{N}(0, \sigma_e^2 I)$, yet our technical results hold for sub-Gaussian noise. Finally, we have the following property on the cost $C_t(\cdot, \cdot)$.

Assumption 2. For any y, y' and u, u' such that $\max\{\| y - y' \|, \| u - u' \|\} \le \Gamma$ and $B_{uy} = \max\{\| y \|, \| u \|\}$, for all $t, |C_t(y, u) - C_t(y', u')| \le R(\| y - y' \|^2 + \| u - u' \|^2)$ and $0 \le C_t(y, u) \le RB_{uy}^2$.

The regret of FALCON is computed with respect to the policy $\pi_\star$ that uses the same MPC policy at each time step with the true transition dynamics F in the control design. Thus, the goal of FALCON is to minimize $\mathbf{Regret}(T) = \sum_{t=1}^{T}(C_t(y_t, u_t) - C_t(y_t^{\pi_\star}, u_t^{\pi_\star}))$. For consistent and reliable initial estimation of the underlying system, we assume that FALCON uses bounded persistently exciting inputs during the warm-up period. Given these inputs, we have the following learning guarantee.

Proposition 3. Let $\alpha_m = \sup_{i, \|s\| \le S} \| \partial^m F_i(s) \|_{L^\infty}$ and $d = h(d_y + d_u)$. Using nth order Fourier basis for learning the model for sufficiently large n, after the warm-up of T_w, with high probability $\sup_{\|s\| \le S} \| F(s) - \hat{F}(s) \|_\infty = \tilde{O}(n^d T_w^{-0.5} + \alpha_m n^{-m})$.

Here $\tilde{O}(\cdot)$ presents the order up to logarithmic terms. The proof is given in Supplementary Material, where we use standard least-squares estimation error analysis and the multivariate analog of Jackson's theorem for trigonometric polynomial approximation[86]. This result shows that the underlying system can be identified with the optimal rate of $1/\sqrt{T}$, yet, due to the properties of the underlying system, there exists a constant term in the estimation error. Note that this constant term depends on the smoothness of the system. For nonlinear systems that live in high-order Sobolev spaces m, this constant term can be small. In the extreme case of infinitely differentiable systems, this constant term approaches to 0. Thus, we have $\sup_{\|s\| \le S} \| F(s) - \hat{F}(s) \|_\infty = \tilde{O}(T_w^{\varepsilon - 0.5})$ after warm-up, where ε depends on the smoothness of the system and the order of Fourier basis. Next, we focus on the adaptive control task. We have the following assumption on the MPC policy that FALCON deploys.

Assumption 4. The MPC policy with $F(\cdot)$ achieves e-IOS, i.e., $\forall t > t_0, \| y_t \| \le (1 - \rho)^{t - t_0} \| y_{t_0} \| + M \sup_{i \in [t_0:t]} \| e_i \|$, for $M > 0$ and $0 < \rho < 1$. The MPC policy with planning model $\hat{F}(\cdot)$, such that $\sup_{\|s\| \le S} \| F(s) - \hat{F}(s) \|_\infty \le \epsilon$, achieves e-IOS with $\rho/2$ and $2M$ and synthesizes persistently exciting inputs which are locally L_o-Lipschitz in planning model.

This assumption states that MPC stabilizes the underlying system by using any model within a neighborhood around the underlying system for planning. This assumption is mild and one can show that it holds for linear systems. Intuitively, this assumption holds for nonlinear systems with valid linearization for bounded inputs. Finally, the last statement allows consistent estimation during the adaptive control with reasonable variations in the input due to model dynamics used in planning. In practice, this condition is usually satisfied by the combination of unmodelled system dynamics, system noise, and sampling-based MPC policies[63]. Note that for a long enough warm-up T_w, the model estimation error can be made small enough to achieve stabilization via the MPC policy. Once FALCON is guaranteed to stabilize the dynamics, it safely regulates the system.

Theorem 5. Suppose Assumptions 1, 2, and 4 hold. Let T_w be chosen long enough such that the MPC policy of FALCON stabilizes the underlying system dynamics. Then, with high probability, for large enough T, FALCON attains regret of $\mathbf{REGRET}(T) = \tilde{O}(\sqrt{T} + \epsilon' T)$, where ϵ' depends on the smoothness of the underlying system. For sufficiently smooth systems, it achieves $\mathbf{REGRET}(T) = \tilde{O}(\sqrt{T})$.

The proof is given in Supplementary Material. This shows that FALCON is the first efficient RL algorithm that attains $\tilde{O}(\sqrt{T})$ regret in adaptive control of partially observable nonlinear systems. Note that this result applies to various systems that are governed by partial differential equations since FALCON only requires the periodic extension of the model dynamics to live in the Sobolev space of periodic functions. Moreover, for infinitely smooth systems, e.g. sinusoidal systems, one can improve Proposition 3, and this in turn would give significantly improved regret upper bound for FALCON.

Corollary 6. Under the setting of Theorem 5, for an infinitely smooth system, i.e., $F_i \in W_p^{\infty,2}([0, 2\pi]^{h(d_y+d_u)})$ for $1 \le i \le d_y$, with high probability, FALCON attains $\mathbf{REGRET}(T) = \mathrm{polylog}(T)$.

This shows that for a certain class of dynamical systems, using the domain knowledge on the system dynamics, FALCON can achieve almost logarithmic regret even if the underlying system is unknown.

Wing system design and manufacturing

The wing system was designed with a standard NACA0012 airfoil shape, which was previously studied for its dynamics in a bluff-body wake at a similar Reynolds number[35,87]. The modular wing body was 3D printed using a combination of materials, allowing for various sensor configurations. The central module, made of micro carbon fiber-filled nylon (Markforged Onyx) reinforced with carbon fibers for added strength and rigidity, housed the primary electronics and secured the system to its mounting via a sweptback fairing. Spanwise sections designed to house individual pressure sensors were also printed using carbon fiber-filled nylon. Clear PLA was used to print the sections between the sensors, which were aligned and connected using carbon fiber spars to add rigidity. Trailing edge flaps were cut from insulation foam and covered with an adhesive-backed coating for protection and improved surface finish.

The wing had a spanwise length of 1 m and a total chord length of 25 cm with 5 cm trailing edge flaps. Using the mean flow velocity near the leading edge, the system had a Reynolds number of approximately $Re \approx 110{,}000$. There were 9 sensor locations distributed symmetrically about the wing, with exactly 10cm between each location. The central sensor location featured a pitot-static tube, with the rest of the sensor locations featuring surface pressure taps. The pressure taps and the pitot-static tube were printed using an SLA printer (Formlabs Form3) for improved surface feature accuracy. Pressure taps were located at 0.4%, 0.7%, 1.5%, and 6% of the chord length from the leading edge on both the pressure and suction sides of the wing. The fairing on which the wing was mounted was reinforced with carbon fiber and aluminum and was set back with an angle of 60° to reduce aerodynamic interactions between the fairing and the wing. The fairing was connected to a set of vertically-aligned air bearings (New Way), which allowed for nearly frictionless motion along a single axis while constraining all other directions. The constrained fairing was mounted directly to a single-axis load cell (Interface SM-50) that passed the signal through an amplifier (Interface Model SGA) with a 50 Hz low pass filter, and the signal was read by a DAQ (NI USB-6008).

The wing had a total of 9 ultra-low range digital pressure sensors (Honeywell RSCDRRM2.5MDSE3) to measure pressure values, which were communicated with a microcontroller (Teensy 4.0). The microcontroller also controlled the high-speed brushless servo motors (MKSHBL6625) that drove the trailing edge flaps. Due to mechanical restraints, the actuation for the servo motors had a maximum/minimum position of $+40°/-40°$. Both the microcontroller and the DAQ communicated with a desktop computer, which received measurements and sent actions. The full control loop operated at approximately 42 Hz.

Generation and characterizations of turbulence

The John W. Lucas Wind Tunnel (LWT) at Caltech was used to conduct all quantitative experiments discussed in this study. The wind tunnel is a closed-loop design with a test section size of 130 cm × 180 cm. To generate turbulence, an asymmetric bluff body was mounted to the wind tunnel using bungee cords, creating a wake of irregular and turbulent flow. The bluff body consisted of a large diameter cylinder (30 cm) with an asymmetrically mounted flat plate at the front, giving the entire body an effective diameter of 53 cm (Fig. 1C). To encourage vortex dislocation and add irregularity to vortex shedding, the cylinder spanned the full width of the tunnel, while the flat plate only had a width of 60 cm. The bluff body was positioned 170 cm upstream of the wing system, with a vertical offset of 48 cm, and sparse elastic bands were placed horizontally across the test section immediately upstream of the bluff body to further increase turbulence intensity.

We used particle image velocimetry (PIV) (Fig. 3) to visualize a limited portion of the bluff body wake. PIV is a quantitative flow visualization technique capable of measuring the velocity fields of fluid flows[88]. Here we performed two-dimensional, two-component (2D2C) PIV. This involves using a laser sheet to illuminate small, dense, neutrally buoyant seed particles. Recording the illuminated particles with a high-speed camera, we can estimate velocity fields by calculating the effective inter-frame displacement of groups of particles via cross-correlation of subsequent images. Performing these experiments in air, we used a 200 mJ/pulse dual pulsed laser (Lumibird Evergreen) to illuminate soap bubbles (15-micron mean diameter) generated with a custom-built bubbler. The flow was recorded with a 4 MP CCD camera (IMPERX Bobcat B2401).

Characterization of the flow near the wing system was performed using a hot-wire anemometer (TSI IFA-300). The anemometer was mounted approximately 2 cm upstream of the leading edge of the airfoil, and measurements were taken at 1000 Hz for 120 s. The turbulence intensity was determined to be 10.8% using the hot-wire anemometer. We found the dominant shedding frequency to be 2.44 Hz, although as mentioned above our oscillating bluff body encouraged irregularities in the shedding process.

Baseline algorithms

In our experiments, besides FALCON, we test several model-free RL methods, including Twin Delayed DDPG (TD3)[55], LSTM-TD3[56], Soft Actor-Critic (SAC)[57], and the industry-standard responsive control strategy of PID controller. Of all these algorithms LSTM-TD3 has recently demonstrated to achieve state-of-the-art performance in predictive flow disturbance rejection[35]. Note that both TD3 and LSTM-TD3 provide deterministic policies, whereas SAC designs stochastic policies. Unlike FALCON, these model-free methods work in episodes where the algorithms stop retraining the policy parameters (reset). We train the model-free methods for 200 episodes of 800 samples per episode and test the best-performing policy in presenting the results. The full 200 episodes of training for the best-performing policy in each method are shown in Fig. 5. All methods were implemented using an NVIDIA GeForce RTX 3070 which enabled a 42 Hz frequency for sensing and control. The brief descriptions of the algorithms are given below with the relevant hyperparameters in Tables 4–7.

TD3 is a deterministic actor-critic type RL framework that builds on previous value-based methods. TD3 injects noise into actions to enable policy exploration (i.e., exploration noise), and injects noise into critic updates to regularize and smooth the resulting policy (i.e., policy smoothing noise). TD3 also uses delayed policy updates which decreases variance in value estimates. For this work, we added gradient clipping to both actor and critic networks to encourage more stable learning in a real-world setting. TD3 has been proven effective in several simulated[55] environments and has previously been used for experimental flow control in a different setting[34]. For further implementation details of TD3 please refer to[55] and the code provided in the submission.

LSTM-TD3 uses the same fundamental algorithm as TD3 but includes LSTM cells for a recurrent actor-critic framework. It was modified from TD3 to better address problems suffering from partial observability[56]. For further implementation details of LSTM-TD3 please refer to[56] and the code provided in the submission.

Table 4 | Hyperparameters of TD3 in our experiments

Hyperparameter	Range	Best
Discount factor	0.95−0.99	0.99
Batch size	16−128	50
Replay buffer size	$2-6\times10^4$	4×10^4
Target update rate	0.005	0.005
Actor learning rate	$10^{-5}-10^{-2}$	10^{-4}
Critic learning rate	$10^{-5}-10^{-2}$	10^{-4}
Exploration noise	0.025−0.2	0.05
Policy smoothing noise	0.025−0.2	0.05
Policy update delay	2−3	3
Target noise clip boundary	0.5	0.5
Actor gradient clip boundary	0.1−1	0.5
Critic gradient clip boundary	0.1−1	0.5

Table 5 | Hyperparameters of LSTM-TD3 in our experiments

Hyperparameter	Range	Best
Discount factor	0.95−0.99	0.99
Batch size	16−128	50
Replay buffer size	$2-6\times10^4$	4×10^4
Target update rate	0.005	0.005
Actor learning rate	$10^{-5}-10^{-2}$	10^{-4}
Critic learning rate	$10^{-5}-10^{-2}$	10^{-4}
Exploration noise	0.025−0.2	0.05
Policy smoothing noise	0.025−0.2	0.05
Policy update delay	2−3	3
Target noise clip boundary	0.5	0.5
Actor gradient clip boundary	0.1−1	0.5
Critic gradient clip boundary	0.1−1	0.5
LSTM Depth	3−15	10

Table 6 | Hyperparameters of SAC in our experiments

Hyperparameter	Range	Best
Discount factor	0.95−0.99	0.99
Batch size	32−256	128
Replay buffer size	$2\times10^4-6\times10^4$	4×10^4
Actor learning rate	$10^{-5}-10^{-2}$	5×10^{-4}
Critic learning rate	$10^{-5}-10^{-2}$	10^{-3}
Target smoothing coefficient	$10^{-3}-10^{-2}$	10^{-2}
Target entropy	−2 to −0.5	−0.5
Temperature learning rate	$10^{-5}-10^{-2}$	5×10^{-4}

Table 7 | Hyperparameters of PID in our experiments

Hyperparameter	Range	Best
K_P	0−15	10
K_I	0−2	0.5
K_D	0−5	2

The Soft Actor-Critic (SAC) method is based on the maximum entropy RL framework that wants to maximize the performance/reward concurrently maximizing the entropy of the policy, i.e., increase the randomness in the policy. Intuitively, this method results in a stochastic policy that achieves good performance and provides the most randomness in achieving this result. SAC uses a temperature parameter to weigh the entropy term relative to the reward function. The ideal temperature parameter can be automatically learned during training. This method is initially proposed for real-world robotic learning to facilitate smooth exploration, tolerate unexpected perturbations/changes during execution, and improve robustness to hyperparameters and sample efficiency. To this end, it requires relatively less hyperparameters compared to other model-free RL methods. For further implementation details of SAC please refer to[57] and the code provided in the submission.

PID control is the most prevalent method found in industrial and commercial applications. PID controllers use a basic feedback control loop that attempts to minimize the error between an observed value and a desired setpoint. PID controllers weigh a proportional term, an integral term, and a derivative term, all of which are tuned for each specific application. As the name suggests, the proportional term contributes a control signal that is directly proportional to the magnitude of the error. The integral term provides a signal that corresponds to the running accumulated error but is slow to react. The derivative term sends a control signal that is proportional to the error rate of change, which effectively smooths behavior. All three of these terms are weighed through corresponding constant values (i.e., K_P, K_I, K_D) that can be found through various tuning methods. For further implementation details of PID please refer to the code provided in the submission.

Data availability

The data and the code used and/or analyzed during the current study are available from the corresponding authors on request.

Received: 24 March 2023; Accepted: 28 August 2024;
Published online: 24 September 2024

References

1. Boettcher, F., Renner, C., Waldl, H.-P. & Peinke, J. On the statistics of wind gusts. *Bound.-Layer. Meteorol.* **108**, 163–173 (2003).
2. Watkins, S. et al. Ten questions concerning the use of drones in urban environments. *Build. Environ.* **167**, 106458 (2020).
3. Kanev, S. & van Engelen, T. Wind turbine extreme gust control. *Wind Energy.: Int. J. Prog. Appl. Wind Power Convers. Technol.* **13**, 18–35 (2010).
4. Shohag, M. A. S., Hammel, E. C., Olawale, D. O. & Okoli, O. I. Damage mitigation techniques in wind turbine blades: A review. *Wind Eng.* **41**, 185–210 (2017).
5. Carcangiu, C., Pujana-Arrese, A., Mendizabal, A., Pineda, I. & Landaluze, J. Wind gust detection and load mitigation using artificial neural networks assisted control. *Wind Energy* **17**, 957–970 (2014).
6. Jones, A. R. Gust encounters of rigid wings: Taming the parameter space. *Phys. Rev. Fluids* **5**, 110513 (2020).
7. Herrmann, B., Brunton, S. L., Pohl, J. E. & Semaan, R. Gust mitigation through closed-loop control. ii. feedforward and feedback control. *Phys. Rev. Fluids* **7**, 024706 (2022).
8. Brunton, S. L. & Noack, B. R. Closed-loop turbulence control: Progress and challenges. *Appl. Mech. Rev.* **67** 05081 (2015).
9. Hou, W., Darakananda, D. & Eldredge, J. D. Machine-learning-based detection of aerodynamic disturbances using surface pressure measurements. *AIAA J.* **57**, 5079–5093 (2019).
10. Pohl, J. E., Radespiel, R., Herrmann, B., Brunton, S. L. & Semaan, R. Gust mitigation through closed-loop control. i. trailing-edge flap response. *Phys. Rev. Fluids* **7**, 024705 (2022).
11. Ventura Diaz, P. & Yoon, S. High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles. In *2018 AIAA Aerospace Sciences Meeting*, 1266 (2018).
12. Han, J. From pid to active disturbance rejection control. *IEEE Trans. Ind. Electron.* **56**, 900–906 (2009).
13. Hau, E. *Wind turbines: fundamentals, technologies, application, economics* (Springer Science & Business Media, 2013).
14. Sontag, E. D. *Mathematical control theory: deterministic finite dimensional systems*, vol. 6 (Springer Science & Business Media, 2013).
15. Hoffmann, G., Huang, H., Waslander, S. & Tomlin, C. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, 6461 (2007).
16. Mellinger, D., Michael, N. & Kumar, V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int. J. Robot. Res.* **31**, 664–674 (2012).
17. Mohamed, A., Clothier, R., Watkins, S., Sabatini, R. & Abdulrahim, M. Fixed-wing mav attitude stability in atmospheric turbulence, part 1: Suitability of conventional sensors. *Prog. Aerosp. Sci.* **70**, 69–82 (2014).
18. Langelaan, J. W., Alley, N. & Neidhoefer, J. Wind field estimation for small unmanned aerial vehicles. *J. Guidance, Control, Dyn.* **34**, 1016–1030 (2011).
19. Wenz, A. & Johansen, T. A. Moving horizon estimation of air data parameters for uavs. *IEEE Trans. Aerosp. Electron. Syst.* **56**, 2101–2121 (2019).
20. Mathisen, S. H., Gryte, K., Johansen, T. & Fossen, T. I. Non-linear model predictive control for longitudinal and lateral guidance of a small fixed-wing uav in precision deep stall landing. In *Aiaa infotech@ aerospace*, 0512 (2016).
21. Bleckmann, H., Mogdans, J. & Coombs, S. L. *Flow Sensing in Air and Water: Behavioral, Neural and Engineering Principles of Operation*. (Springer, 2014).
22. Triantafyllou, M. S., Weymouth, G. D. & Miao, J. Biomimetic survival hydrodynamics and flow sensing. *Annu. Rev. Fluid Mech.* **48**, 1–24 (2016).
23. Elder, J. & Coombs, S. The influence of turbulence on the sensory basis of rheotaxis. *J. Comp. Physiol. A* **201**, 667–680 (2015).
24. Sterbing-D'Angelo, S. et al. Bat wing sensors support flight control. *Proc. Natl Acad. Sci.* **108**, 11291–11296 (2011).
25. Mohamed, A., Abdulrahim, M., Watkins, S. & Clothier, R. Development and flight testing of a turbulence mitigation system for micro air vehicles. *J. Field Robot.* **33**, 639–660 (2016).
26. Meier, L. et al. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots* **33**, 21–39 (2012).
27. Gavrilovic, N., Bronz, M., Moschetta, J.-M. & Bénard, E. Bioinspired wind field estimation-part 1: Angle of attack measurements through surface pressure distribution. *Int. J. Micro Air Veh.* **10**, 273–284 (2018).
28. Krieg, M., Nelson, K. & Mohseni, K. Distributed sensing for fluid disturbance compensation and motion control of intelligent robots. *Nat. Mach. Intell.* **1**, 216–224 (2019).
29. Shi, X. et al. Adaptive nonlinear control of fixed-wing vtol with airflow vector sensing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 5321–5327 (IEEE, 2020).
30. O'Connell, M. et al. Neural-fly enables rapid learning for agile flight in strong winds. *Sci. Robot.* **7**, eabm6597 (2022).
31. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).
32. Bieker, K., Peitz, S., Brunton, S. L., Kutz, J. N. & Dellnitz, M. Deep model predictive flow control with limited sensor data and online learning. *Theor. Computational Fluid Dyn.* **S.I.**, 577–591 (2020).
33. Gunnarson, P. et al. Learning efficient navigation in vortical flow fields. *Nat Commun* **12**, 7143 (2021).
34. Fan, D., Yang, L., Wang, Z., Triatafyllou, M. S. & Karniadakis, G. E. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proc. Natl Acad. Sci.* **117**, 26091–26098 (2020).

35. Renn, P. I. & Gharib, M. Machine learning for flow-informed aerodynamic control in turbulent wind conditions. *Commun. Eng.* **1**, 1–9 (2022).
36. Jaksch, T., Ortner, R. & Auer, P. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.* **11**, 1563–1600 (2010).
37. Littman, M. L., Cassandra, A. R.. & Kaelbling, L. P. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning.* 362–370 (1995).
38. Azizzadenesheli, K., Lazaric, A. & Anandkumar, A. Reinforcement learning of pomdps using spectral methods. In *29th Annual Conference on Learning Theory.* **49**, 193–256 (2016).
39. Nagabandi, A., Konolige, K., Levine, S. & Kumar, V. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, 1101–1112 (PMLR, 2020).
40. Pope, S. B.*Turbulent Flows* (Cambridge university press, 2000).
41. Mueller, T. J. & DeLaurier, J. D. Aerodynamics of small vehicles. *Annu. Rev. fluid Mech.* **35**, 89–111 (2003).
42. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc.: Ser. B (Methodol.)* **58**, 267–288 (1996).
43. Bearman, P. W. On vortex shedding from a circular cylinder in the critical reynolds number regime. *J. Fluid Mech.* **37**, 577–585 (1969).
44. Taira, K. et al. Modal analysis of fluid flows: An overview. *AIAA J.* **55** 4013–4041 (2017).
45. Hussaini, M. Y. & Zang, T. A. Spectral methods in fluid dynamics. *Annu. Rev. fluid Mech.* **19**, 339–367 (1987).
46. Khalil, H. *Nonlinear Systems*, vol. 3 (Prentice Hall, 2002).
47. Chen, W.-H., Ballance, D. J. & Gawthrop, P. J. Optimal control of nonlinear systems: a predictive control approach. *Automatica* **39**, 633–641 (2003).
48. Erez, T. et al. An integrated system for real-time model predictive control of humanoid robots. In *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*, 292–299 (IEEE, 2013).
49. Botev, Z. I., Kroese, D. P., Rubinstein, R. Y. & L'Ecuyer, P. The cross-entropy method for optimization. In *Handbook of statistics*, vol. 31, 35–59 (Elsevier, 2013).
50. Williams, G. et al. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1714–1721 (IEEE, 2017).
51. Schoenwald, D. A. System identification using a wavelet-based approach. In *Proceedings of 32nd IEEE Conference on Decision and Control*, 3064–3065 (IEEE, 1993).
52. Roshko, A. On the wake and drag of bluff bodies. *J. aeronautical Sci.* **22**, 124–132 (1955).
53. von Kármán, T. Collapse of the tacoma narrows bridge. *Resonance* **10** 97–102 (2005).
54. Williamson, C. K. vortex dynamics in the cylinder wake. *Annu. Rev. Fluid Mech.* **28**, 477–539 (1996).
55. Fujimoto, S., Hoof, H. & Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596 (PMLR, 2018).
56. Meng, L., Gorbet, R. & Kulić, D. Memory-based deep reinforcement learning for pomdps. *IROS 2021* (2021).
57. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning,* 1861–1870 (2018).
58. Garnier, P. et al. A review on deep reinforcement learning for fluid mechanics. *Computers Fluids* **225**, 104973 (2021).
59. Haarnoja, T. et al. Learning to walk via deep reinforcement learning. In *Proceedings of Robotics: Science and Systems* (2019).
60. Vlachas, P. R. et al. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191–217 (2020).
61. Wang, W., Yu, N., Gao, Y. & Shi, J. Safe off-policy deep reinforcement learning algorithm for volt-var control in power distribution systems. *IEEE Trans. Smart Grid* **11**, 3008–3018 (2019).
62. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations* (2020).
63. Lale, S., Azizzadenesheli, K., Hassibi, B. & Anandkumar, A. Model learning predictive control in nonlinear dynamical systems. *2021 IEEE 60th Conference on Decision and Control (CDC)* (2021).
64. Nocedal, J. & Wright, S. J. *Numerical optimization* (Springer, 1999).
65. Li, W. C. & Biegler, L. T. A multistep, newton-type control strategy for constrained, nonlinear processes. In *1989 American Control Conference*, 1526–1527 (IEEE, 1989).
66. Huang, K., Lale, S., Rosolia, U., Shi, Y. & Anandkumar, A. Cem-gd: Cross-entropy method with gradient descent planner for model-based reinforcement learning. arXiv preprint arXiv:2112.07746 (2021).
67. Leontaritis, I. & Billings, S. A. Input-output parametric models for non-linear systems part ii: stochastic non-linear systems. *Int. J. control* **41**, 329–344 (1985).
68. Boyd, S. & Chua, L. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Trans. circuits Syst.* **32**, 1150–1161 (1985).
69. Dahleh, M. A., Sontag, E. D., David, N. & Tsitsiklis, J. N. Worst-case identification of nonlinear fading memory systems. *Automatica* **31**, 503–508 (1995).
70. Canuto, C., Hussaini, M. Y., Quarteroni, A. & Zang, T. A.*Spectral methods: fundamentals in single domains* (Springer Science & Business Media) 2007
71. Yang, Y. et al. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, 1-10 (PMLR, 2020).
72. Iscen, A. et al. Policies modulating trajectory generators. In *Conference on Robot Learning*, 916–926 (PMLR, 2018).
73. Sherman, J. & Morrison, W. J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Stat.* **21**, 124–127 (1950).
74. Oymak, S. & Ozay, N. Non-asymptotic identification of lti systems from a single trajectory. In *American Control Conference (ACC),* 5655–5661 (2019).
75. Simchowitz, M., Mania, H., Tu, S., Jordan, M. I. & Recht, B. Learning without mixing: Towards a sharp analysis of linear system identification. In *Proceedings of the 31st Conference On Learning Theory,* 439–473 (PMLR, 2018).
76. Tsiamis, A., Matni, N. & Pappas, G. J. Sample complexity of kalman filtering for unknown systems. In *Learning for Dynamics and Control,* 435–444 (PMLR, 2019).
77. Faradonbeh, M. K. S., Tewari, A. & Michailidis, G. On adaptive linear–quadratic regulators. *Automatica* **117**, 108982 (2020).
78. Cohen, A., Koren, T. & Mansour, Y. Learning linear-quadratic regulators efficiently with only $\sqrt{T}$ regret. In *Proceedings of the 36th International Conference on Machine Learning,* 1300–1309 (PMLR, 2019).
79. Simchowitz, M. & Foster, D. J. Naive exploration is optimal for online lqr. In *Proceedings of the 37th International Conference on Machine Learning*, 8937–8948 (PMLR, 2020).
80. Lale, S., Azizzadenesheli, K., Hassibi, B. & Anandkumar, A. Logarithmic regret bound in partially observable linear dynamical systems. *Adv. Neural Inf. Process. Syst.* **33**, 20876–20888 (2020).
81. Chen, X. & Hazan, E. Black-box control for linear dynamical systems. In *Proceedings of Thirty Fourth Conference on Learning Theory*, 1114–1143 (PMLR, 2021)
82. Lale, S., Azizzadenesheli, K., Hassibi, B. & Anandkumar, A. Reinforcement learning with fast stabilization in linear dynamical systems. In *International Conference on Artificial Intelligence and Statistics*, 5354–5390 (PMLR, 2022).

83. Sattar, Y. & Oymak, S. Non-asymptotic and accurate learning of nonlinear dynamical systems. *Journal of Machine Learning Research* **23–140**, 1–49 (2022).
84. Mania, H., Jordan, M. I. & Recht, B. Active learning for nonlinear system identification with guarantees. *Journal of Machine Learning Research* **23–32**, 1–30 (2022).
85. Kakade, S., Krishnamurthy, A., Lowrey, K., Ohnishi, M. & Sun, W. Information theoretic regret bounds for online nonlinear control. *Adv. Neural Inf. Process. Syst.* **33**, 15312–15325 (2020).
86. Schultz, M. H. L^{∞}-multivariate approximation theory. *SIAM J. Numer. Anal.* **6**, 161–183 (1969).
87. Lefebvre, J. N. & Jones, A. R. Experimental investigation of airfoil performance in the wake of a circular cylinder. *AIAA J.* **57**, 2808–2818 (2019).
88. Willert, C. E. & Gharib, M. Digital particle image velocimetry. *Exp. fluids* **10**, 181–193 (1991).

Acknowledgements

The authors would like to thank Professor John Dabiri (Caltech) for the insightful discussion regarding this work. Anima Anandkumar is Bren chair and Schmidt sciences AI 2050 senior fellow and this work was supported by the Center for Autonomous Systems and Technologies at Caltech, the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1745301, the US Office of Naval Research (MURI grant N00014-18-12624).

Author contributions

All the authors conceptualized and planned the testing methodology together. S.L. and P.R. were responsible for the algorithm design, prototype design, manufacturing, data acquisition, programming, visualizations, theoretical derivations, and writing the original paper drafts. K.A., B.H., M.G., and A.A. supervised and provided advice throughout the entire project, and helped edit and review the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44182-024-00013-0.

Correspondence and requests for materials should be addressed to Anima Anandkumar.