Deep Reinforcement Learning for Fault Adaptive Control

Luke Bhan[1] , **Marcos Quinones-Grueiro**[1] , **Gautam Biswas**[1]
[1]Vanderbilt University, Nashville TN, 37209 USA
{luke.bhan, marcos.quinones.grueiro, gautam.biswas}@vanderbilt.edu

Abstract

The degradation of a system property or parameter reduces the performance of the control system sometimes causing a complete breakdown. Fault adaptive control focuses on developing algorithms to mitigate the impact of system faults on the control task performance. Deep reinforcement learning (DRL) methods have demonstrated outstanding performance for continuous control tasks. However, faults affect the system dynamics violating the Markov property such that optimality guarantees are lost. We propose an adaptive control scheme based on a combination of DRL and classic control to achieve fault tolerance. We empirically demonstrate our approach on an octocopter case study subject to motor faults. We compared on-policy and off-policy algorithms for the adaptive control task. Results showed on-policy agents outperforming off-policy agents for the trajectory-tracking problem.

1 Introduction

Fault Adaptive Control strategies address the problem of improving the control performance when systems operate in a degraded manner due to faults [Blanke *et al.*, 2016]. Classic fault adaptive methods require an accurate and comprehensive dynamic model of the system. Data-driven methods like DRL, on the other hand, allow developing control strategies by using data. However, DRL methods for direct control lose convergence guarantees for fault adaptive problems. Recent approaches seek therefore to combine classic control with DRL to leverage the best of both worlds [Fei *et al.*, 2020; Sohège *et al.*, 2020]. We propose in this work a novel approach that combines parameter estimation techniques with classic control and DRL to solve the fault adaptive control problem. We compare the effect of different learning algorithms on an octocopter trajectory tracking task subject to a motor fault with varying magnitude.

2 Deep Reinforcement Learning

Reinforcement learning (RL) methods aim to solve an optimal control problem through learning methods. The control law is refined through the continuous interactions of a learning agent with an environment [Sutton and Barto, 2018]. The learning problem is formalized through the Markov Decision Process defined by a four tuple: $M = \{S, A, T, R\}$, where S represents the set of possible states in the environment. The transition function $T : S \times A \times S \rightarrow [0, 1]$ defines the probability of reaching state s' at $t + 1$ given that action $a \in A$ was chosen in state $s \in S$ at *decision epoch* t, $T = p(s'|s, a) = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}$. The reward function $R : S \times A \rightarrow \Re$ estimates the immediate reward $R \sim r(s, a)$ obtained from choosing action a in state s. The objective of the agent is to find an optimal policy π^* that maximizes the following criteria for $\forall s \in S$:

$$V^{\pi^*}(s) = \max_{\pi \in \Pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_t = \pi(s_t)\right], \tag{1}$$

where $V^\pi : S \rightarrow R$ is called value function and $0 < \gamma \leq 1$ is called the discount factor, and it determines the weight assigned to future rewards. The agent's objective is to find the policy that maximizes the expected sum of reward. Obtaining a policy with optimality guarantees requires the following two conditions to be satisfied

1. $|R \sim r(s, a)| \leq C < \infty, \forall a \in A, s \in S$
2. T and R do not change over time.

Systems subjects to faults undergo changes that cause their dynamic model, represented by the transition function T, to change over time [Dulac-Arnold *et al.*, 2019]. Therefore, learning direct control with DRL for fault tolerance is not theoretically feasible. We propose a fault adaptive control scheme that avoids using DRL for direct control in the next section.

3 Proposed Fault Adaptive Control Scheme

We propose the combination of classic control schemes with DRL as a solution to the Fault Adaptive Control problem presented in Figure 1. Classic control methods like Proportional Integral Derivative (PID) Controllers remain dominant in real world industry applications thanks to their simple structure, ease of implementation, and wide variety of tuning methods [Borase *et al.*, 2021]. However, traditional tuning methods for PID control do not account simultaneously for multiple input-output systems and multiple PID controllers. Carlucho

et al. [2020] propose to use DRL for PID parameter tuning to tackle the previously mentioned problems in robotic tasks. Moreover, adaptation is required for control systems which undergo through faults. Hence we extend the scheme proposed in [Carlucho *et al.*, 2020] to accommodate faults assuming they are not catastrophic (the system can continue to operate in a degraded manner and implying that the PID controller updates its parameters to recover performance. The

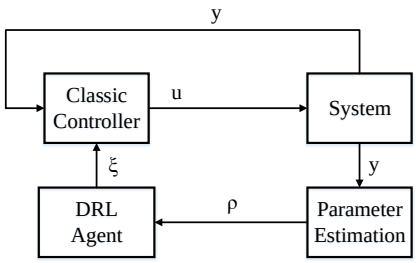

Figure 1: Fault Adaptive Control framework

core of the proposed approach relies in the combination of parameter estimation techniques with DRL. We update the PID controller when the value of the system parameter(s) associated with faults degrades the control task performance. The measurements obtained from the system $y \in \Re^m$ are used to estimate fault-related parameters $\rho \in \Re^n$ through estimation techniques like the Unscented Kalman Filter and the Particle Filter [Daigle *et al.*, 2012]. The estimated parameters are used as inputs to the DRL agent ($s = \rho$) and the action of the agent consists in a new set of parameters for the controller(s) ($a = \xi$). We demonstrate the feasibility of the proposed approach in a complex control task presented next.

4 Octocopter control task

We consider an octocopter airframe dynamics model based on Newton-Euler equations of motion for a rigid body [Powers *et al.*, 2015]. The octocopter's cascade control scheme is shown in Figure 2. This control approach allows for stabilization of the position and orientation of the octocopter with respect to a trajectory. A set of three PID controllers adjust the vehicle attitude, and different three PID controllers adjust the position variables, together forming nested feedback loops. The

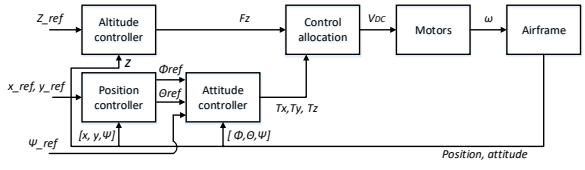

Figure 2: Cascade Control scheme for the octocopter

reference trajectory is defined in terms of position and yaw angle $[x_t, y_t, z_t, \psi_t]$. The Altitude PID controller generates the required force in the z direction. The position PD controllers estimate, based on the current position of the vehicle and yaw angle, the reference for the pitch (θ) and roll (ϕ) angles. The attitude PD controllers generate the required torque in each direction. The control allocation block transforms the torques and force into a reference voltage for each motor of the octocopter. Finally, each motor generates angular velocity according to the motor's dynamics. More details of the octocopter modeling and control allocation can be found in [Quinones-Grueiro *et al.*, 2021].

4.1 Fault scenarios

Typically, the degradation of the components of the octocopter increases monotonically from mission to mission. Motors are susceptible to mechanical degradation in the form of bearing wear, and electrical degradation in the form of contact corrosion and insulation deterioration [Abramov *et al.*, 2014]. Instead of generating faults through the manipulation of control signals as it has been done in previous works, we take a more realistic simulation approach and generate the faults by modifying the value of the motor parameters. An increase in winding resistance results in the loss of effectiveness of the motor. Therefore, through the modification of this parameter we generate faulty behaviors of the octocopter in the trajectory-tracking task. In this work, we considered single motor faults ranging between 3 and 8 times the nominal value of the resistance of the motor shown in figure 3.

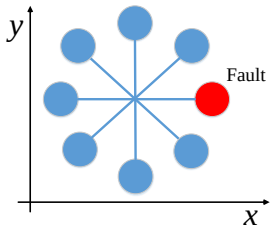

Figure 3: Octocopter motor fault configuration

4.2 DRL training approach and conditions

In this work, we considered training the DRL agent to learn how to adapt the parameters of PD position controller, this implies a four-dimensional action space $a = \{K_p^x, K_d^x, K_p^y, K_d^y\}$. The only information received by the agent is the motor resistance estimated and the reward function is defined by $R = (10 - error)/10$ where $error$ is the Euclidean distance calculated between the position of the octocopter and the reference trajectory. We defined 10 meters as the maximum deviation allowed from the reference trajectory and we re-scale the reward between 0-1 as suggested for continuous control tasks[Henderson *et al.*, 2018].

We considered a change in the reference from $(x = 0, y = 0)$ to $(x = 5, y = 5)$ as the trajectory tracking task for simplification purposes and assuming that the resulting architecture will scale well as long as the changes in the reference are smaller than the one experience during training. Different fault magnitudes must be experienced by the agent to learn how to adapt the position controller parameters. However, we noticed that randomly selecting the fault magnitude for

each episode resulted in no convergence. We thus defined a curriculum learning approach where we first expose the agent to the lower bound of the fault magnitude until it converges and then we generate for each episode with probability of 0.5 a fault with maximum magnitude. In this way, we avoid the catastrophic forgetting problem for the agent.

We compare the performance of four different DRL algorithms for learning to solve the presented Fault Adaptive Control task: Proximal Policy Optimization [Schulman *et al.*, 2017], Deep Deterministic Policy Gradient [Lillicrap *et al.*, 2015], Twin-Delayed DDPG [Fujimoto *et al.*, 2018], and Soft Actor Critic [Haarnoja *et al.*, 2018]. The first one is an on-policy method while the last three are off-policy methods with last two being improvements of DDPG. We performed this experiment with the goal of determining which type of method is more suitable for the specific task to be solved. We define other relevant training hyper-parameters in Table 1. For PPO we considered the same optimizer, discount factor, and episodes configuration but with an initial learning rate of 0.0001. We tuned the neural network hyper-parameters with grid-search approach and found the best architecture to be the same for the actor and the critic with 2 layers and 64 neurons in each one. We clarify that an agent updates its parameters only after a complete episode to understand the effects of the selected PD parameters along the trajectory tracking task. Finally, we do not explore in this paper the effect of biased parameter estimations on the task performance given the space constraints.

Parameter	Value
Discount factor	0.99
Optimizer	ADAM
Initial learning rate	0.0001
Batch size	100
Replay buffer size (time-steps)	100000
Time steps of an episode	2000
Number of episodes	3000

Table 1: DDPG, SAC, and TD3 training parameters

5 Results

We show the training results over ten trials in Figure 4. As it is shown, the agents trained with the four algorithms asymptotically converge. The agents first converge for the lower-bound fault magnitude around 1000 episodes. Once the upper-bound fault magnitude is introduced, the reward per episode decays significantly for all agents. We observed that both PPO and SAC achieved a better performance in terms of learning speed and asymptotic performance. Figures 5 and 6 show the absolute trajectory-tracking error during a complete episode for each axis and a fault magnitude of 8 times the nominal value of the resistance. It is clear from this figure that the final agents obtained through the PPO learning method achieve best performance in terms of mean and variance. We attribute the poor performance of off-policy methods to the use of the replay buffer. While algorithms like PPO guarantee monotonic improvement, this is not the case for the off-policy

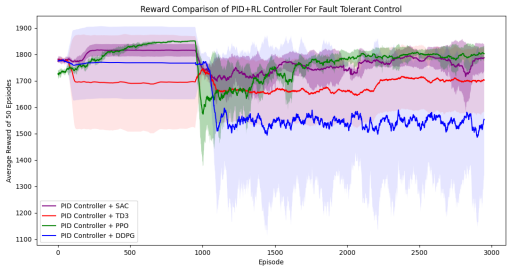

Figure 4: Reward comparison for different DRL algorithms

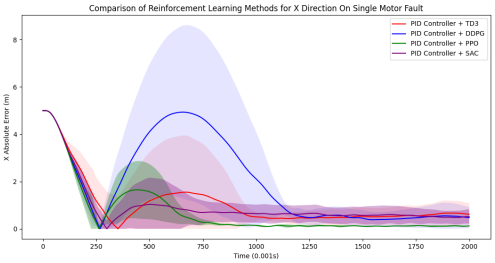

Figure 5: Testing error in the X direction

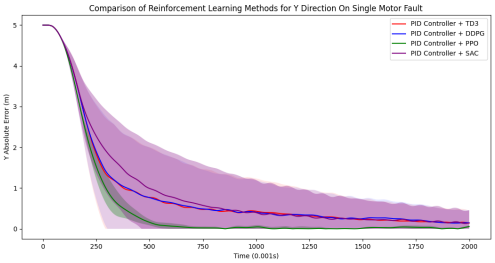

Figure 6: Testing error in the Y direction

methods tested. We consider that new sampling strategies from the replay buffer have to be designed for training fault adaptive control agents.

6 Conclusions

We presented a Fault Adaptive Control architecture combining DRL and classic control methods. We tested the proposed framework with an octocopter considering a cascade control scheme subject to faults in one of the motors. The position controllers are adapted according to the fault magnitude estimated. We compared on-policy and off-policy learning algorithms to train the DRL agent and found that the former are more suitable for the fault tolerant task. As such, this work provides a framework to modelling the complex Octorotor dynamics when faced with a fault scenario and can be applied in real-environments by initiating the fault based DRL agent when a fault reaches a certain magnitude. Therefore, future works will c extend the presented experiments to multiple faults as well as a robustness analysis to understand

the effect of biased parameter estimations on the task performance in both simulated and real-world Octocopter scenarios.

References

[Abramov *et al.*, 2014] I Abramov, Y Nikitin, A Abramov, E Sosnovich, and P Bozek. Control and diagnostic model of brushless dc motor. *Journal of Electrical Engineering*, 65(5), 2014.

[Blanke *et al.*, 2016] Mogens Blanke, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 2016.

[Borase *et al.*, 2021] Rakesh P Borase, D K Maghade, S Y Sondkar, and S N Pawar. A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2):818–827, 2021.

[Carlucho *et al.*, 2020] Ignacio Carlucho, Mariano De Paula, and Gerardo G Acosta. An adaptive deep reinforcement learning approach for MIMO PID control of mobile robots. *ISA Transactions*, 102:280–294, 2020.

[Daigle *et al.*, 2012] Matthew Daigle, Bhaskar Saha, and Kai Goebel. A comparison of filter-based approaches for model-based prognostics. In *2012 IEEE Aerospace Conference*, pages 1–10, 2012.

[Dulac-Arnold *et al.*, 2019] Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019.

[Fei *et al.*, 2020] F. Fei, Z. Tu, D. Xu, and X. Deng. Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7358–7364, 2020.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[Henderson *et al.*, 2018] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[Lillicrap *et al.*, 2015] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. sep 2015.

[Powers *et al.*, 2015] Caitlin Powers, Daniel Mellinger, and Vijay Kumkar. Quadcopter kinematics and dynamics. *Handbook of Unmanned Ariel Vehicles*, 2015.

[Quinones-Grueiro *et al.*, 2021] M Quinones-Grueiro, G Biswas, I Ahmed, T Darrah, and C Kulkarni. Online decision making and path planning framework for safe operation of unmanned aerial vehicles in urban scenarios. *International Journal of Prognostics and Health Management*, 12(3), 2021.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sohège *et al.*, 2020] Yves Sohège, Gregory Provan, Marcos Quiñones-Grueiro, and Gautam Biswas. Deep Reinforcement Learning and Randomized Blending for Control under Novel Disturbances. *IFAC-PapersOnLine*, 53(2):8175–8180, 2020.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.