Policy reuse for transfer reinforcement learning in fault-tolerant control

Ibrahim Ahmed[1] and **Marcos Quinones-Grueiro**[2] and **Gautam Biswas**[3]
Vanderbilt University *
[1]ibrahim.ahmed@vanderbilt.edu
[2]marcos.quinones.grueiro@vanderbilt.edu
[3]gautam.biswas@vanderbilt.edu

Abstract

This paper proposes transfer learning in a reinforcement learning framework for fault-tolerant control of a class of dynamical systems. Our approach adapts policy-reuse algorithms to achieve learning speed improvement when new faults occur in the system. The policy-reuse strategy finds a policy parameter initialization by sampling learned policies for past fault occurrences. We present the conditions under which our approach becomes effective and empirically demonstrate our approach on a test-bed of a 6-tank fuel transfer system of an aircraft.[1].

1 Introduction

Physical systems operating in the real world are subject to degradation and faults during operation. It is important for the system to respond to these changes in a way that it continues to operate, be it in a degraded manner to avoid catastrophic failures. Such fault-tolerant control (FTC) [4] approaches can reduce the design cost of systems by relaxing the constraints on designers to make a system completely fail-safe and improve safety during operations. FTC seeks to keep a faulty system operating within acceptable margins of sub-optimal performance, and this allows for considering trade-offs between design and operating costs.

Data-driven approaches to FTC [10, 9] exploit the preponderance of data collected from system operations. They generate models that avoid the need for time-consuming and accurate physics-based simulations of system dynamics to respond to different situations that may occur in the system. However, such methods depend on the data to span the breadth of operating conditions, and the model has to contain sufficient detail to capture multiple operating modes and faulty situations. This represents another compromise between design and operating costs.

In many cases, systems are complex, the number of possible faults are large, and faults that have not been seen before can occur during operations. Therefore, there is no available system operational data to model such behaviors, and data-driven approaches cannot learn a sufficiently optimal control policy to address such behaviors. Reinforcement learning (RL) presents a semi-supervised approach by forfeiting the dependence on labeled ground truth, and instead relying on accumulated feedback (i.e., experience gained) from a sequence of actions to converge to a globally optimal policy over time. This ability to learn during operations alleviates design time effort and costs. Deep RL methods use complex, nonlinear approximations of the value function to overcome the computational intractability of the problem [5, 2]. However, the dependence on data to learn such approximations limits how fast and how accurately a RL-based controller can adjust to faults.

In ([1]), the authors developed data-driven models to supplement experience with the real environment when known and unknown faults occur in a system. In this work, we develop a transfer learning framework for faster adaption of the RL policy parameters to collected data samples. Our approach is not dependent on the time-consuming step of learning a data-driven model first. Instead, it reuses the results of previous learning to quickly respond to a fault and adapt to a new optimum in due time.

The paper is organized as follows. Section 2 gives a background including extant research (section 2.1), and theoretical basis for the proposed approach (section 2.2). The transfer learning approach is documented in section 3. Finally, simulation results are shown in section 4.

2 Background

2.1 Existing work

Reinforcement learning (RL) has demonstrated promising results when applied to continuous process control across different industries [12]. One shortcoming of reinforcement learning algorithms for continuous control is its sample inefficiency. A strategy to address this issue is to pre-train the policy with a simulator and combine them with domain randomization techniques to tackle the sim2real transfer problem. Fault tolerant control applications pose an even more challenging problem because all possible fault configurations cannot be simulated/randomized for most systems. Therefore, re-learning is a requirement once a fault occurs.

In recent work, RL has been applied for fault tolerant control problems under different assumptions. A stabilizing controller is trained in [8] to maintain position control of an unmanned aerial vehicle regardless of whether a fault/attack is present or not. They assume that a single controller can

*This work was partially supported by NASA SWS Award number 80NSSC21M0087 (subaward 21-S06 to Vanderbilt University)

[1]The code for this work can be obtained at `https://git.isis.vanderbilt.edu/ahmedi/airplanefaulttolerance`

be trained to perform well across all faults considering that fault magnitudes are pre-defined within a given range. A similar approach is presented in [13] where the physical parameters are randomized during learning to obtain a policy robust to faults. Zhang and Gao (2020) propose to accelerate the training process when a fault occurs by introducing a supervisory learning on the basis of a training data-set with particle swarm optimization as learning algorithm. This approach resembles the learning from demonstrations techniques commonly used in transfer learning problems [18].

Faults can be broadly classified into incipient and abrupt. Incipient faults account for degradation mechanisms that slowly change the behavior of the system. Lifelong learning approaches can be adopted to tackle incipient faults [1]. However, abrupt faults alter the system dynamics model, and, therefore, present a new task to be solved, We frame the re-learning process required when an abrupt fault occurs as a transfer learning problem. We learn how to solve a new fixed task by combining observed experiences after the fault occurs with the knowledge from previously solved tasks [18]. While transfer learning methods have been proposed recently for fault detection in supervised learning settings [6], to the best our knowledge, transfer learning has not been applied to solve the abrupt fault tolerant control problem.

Transfer learning in RL settings has been used to speed up the learning process when interactions with the environment are costly by reward shaping, learning from demonstrations, and policy transfer (policy distillation and policy reuse) [18]. In this work, we combine ideas from policy reuse strategy to develop a transfer learning framework for abrupt fault tolerant control.

2.2 Preliminaries

Reinforcement learning

Reinforcement learning (RL) is a semi-supervised machine learning approach. It relies on a controller interacting with an environment which yields feedback: a reward signal. The optimization objective is to select control actions to maximize cumulative rewards over time. The function that selects control actions is called a policy π.

A RL task can be represented by a Markov Decision Process (MDP). An MDP consists of states (x), actions (u), a reward function, $r_t \leftarrow R(x_t, x_{t+1})$, and a state transition function, $x_{t+1} \leftarrow T(x_t, u_t)$. The functions can be stochastic. Using these, the optimal action at time $t = \tau$ becomes:

$$u_\tau \leftarrow \arg\max_u \left(r_\tau + \sum_{t=\tau+1}^{\inf} \gamma^{t-\tau} r_t \right) \tag{1}$$

Where $\gamma \in \mathbb{R}[0, 1]$ is a discount factor to prioritize immediate rewards. The cumulative rewards of optimal actions proceeding from a state are its value $V(x)$. Equation (1) is recursive an can be solved via dynamic programming, as first introduced by [3]. Later improvements such as Q-Learning [16] iteratively tabulated the cumulative rewards (i.e. values) of actions to then derive the most rewarding action. Later still, value-function approximations were used with the help of neural networks [11] to great success. This process of estimating state and action values so the most valuable one can be picked is called policy iteration.

Policy gradient approaches [15] directly iterate over a policy function $u \leftarrow \pi_\theta(x)$ parametrized by θ. They bypass the need of explicit value function approximation to evaluate each state. This is specially useful for continuous action spaces where the controller does not have to search for the action with the highest value. Proximal Policy Optimization (PPO) [14] is one such approach. Internally, it has a *critic* function which evaluates states and actions. Those valuations are then used to train the *actor* function which maps states to action probabilities i.e. the policy. PPO takes care not to change the policy drastically with each iteration of the optimization process. PPO is employed as the RL algorithm of choice in this work.

Transfer learning

Transfer learning methods [19] are designed to automatically build prior knowledge from the solution of a set of source tasks (i.e., training tasks) to be used during the learning process on a new task (i.e., testing task). The idea is to retain and reuse the knowledge across different but related tasks to improve the learning performance.

Formally, we define a RL task $M \sim \Omega$ as a MDP, where Ω represents the distribution from the available space of tasks. The goal of a transfer learning algorithm is to extract knowledge from a set of L source tasks to improve the learning process and/or performance on a target task M_t.

Typically there are three performance metrics considered for transfer learning problems: jump-start improvement, asymptotic improvement, and learning speed improvement. The first one measures the initial performance of a policy compared to random initialization. The second one measures the improvement of the final performance achieved by the policy. The third one measures the efficiency of learning by reducing the required interactions with the environment.

Problem formulation

The FTC challenge is posed as a transfer learning problem. Given a history of control policies learned from other tasks in the distribution (faults), and sampled observations from a new task (fault), find the policy expected to yield the highest feedback on that task.

3 Approach

The proposed FTC scheme takes over after an abrupt fault is diagnosed in the system. It maintains a library of policy parameters previously learned under different tasks. On encountering a new fault, observations are buffered. A policy expected to generate larger rewards on that buffer is sampled from the library and substituted into the controller. Once initialized, the controller starts interacting with the new task and fine-tunes policy parameters according to the RL algorithm being used. The trained parameters are then consolidated in the library for future reuse.

3.1 Discriminating between policies

The choice to maintain a library of policies can be justified by the diverse modes of operations they cover. The policies change substantially with the learning task for each fault. The policies being randomly initialized and highly non-linear functions, their parameter distances cannot be used directly to measure similarity. Policies may also behave differently when subjected to two different tasks. So performance difference on a nominal task is not necessarily the same difference on a separate task.

Therefore, similarity between two policies π_1, π_2 is computed dynamically using probabilities of actions (p_1, p_2) extracted from a buffer $\mathcal{M}$ of states encountered and actions taken by the controller on the target task M_t. One such measure is the Jensen-Shannon divergence, $JSD(\pi_1 \| \pi_2)$, [7], which is based on Kullback-Leibler (KL) divergence between two probability distributions $D(\pi_1 \| \pi_2)$. The divergence calculates the similarity between two probability distributions. In this case the distributions are the action probabilities of states in $\mathcal{M}$. In other words, the controller is asking the question: which policy in the library behaves most similarly under the same fault as the extant policy?

$$\begin{aligned} \vec{p_{1,2}} &= \{\pi_{1,2}(x,u) : x,u \in \mathcal{M}\} \\ \vec{p_M} &= (\vec{p_1} + \vec{p_2})/2 \\ JSD(\pi_1 \| \pi_2) &= \left(D(\vec{p_1} \| \vec{p_M}) + D(\vec{p_2} \| \vec{p_M})\right)/2 \end{aligned} \quad (2)$$

Where p_M is the mixture of the two compared distributions. JS divergence is symmetric, unlike KL divergence. Therefore $JSD(\pi_1 \| \pi_2) \equiv JSD(\pi_2 \| \pi_1)$. The square root of JSD can be used as a distance metric. Policies which produce a smaller distances are likely to behave similarly and thus produce the same feedback. Such policies can be clustered or filtered out, to reduce the sample size in the library. Library pruning is left as a discussion for a later work.

3.2 Ranking optimal policies

Policies previously learned are ranked by how well they can be expected to perform under the new task at initialization. The policy estimated to yield this largest cumulative rewards is selected by the controller as optimal. The architecture of policy gradient algorithms like PPO is leveraged to sample optimal policies. PPO's policy is trained as a *critic* function that evaluates actions taken from states $V(x,u)$, in turn to train the *actor* function that outputs the probability of actions to take $P(u)$.

Let the current nominal task be M, and the controller policy be π. Let there be a library of policies trained on tasks (faults) previously encountered by the controller $\Pi = \{\pi_l : l \in L\}$. When a new fault occurs, the controller faces a new test task $M_t \sim \Omega$, which would have an optimal choice of π_{l^*} from Π. Let $\mathcal{M}_l$ be the buffer of experiences had a library policy π_l been used to collect observations in the aftermath of a fault. Whereas $\mathcal{M}$ is the buffer using the extant policy π. Then the selection of the optimal policy initialization is given by equation 3.

$$\begin{aligned} V(\mathcal{M}_l)_{\pi_l} &= \sum_{x,u \sim \mathcal{M}_l} V(x,u) \cdot \pi_l(x,u) \\ \pi_{l^*} &= \arg\max_{\pi_l \sim \Pi} V(\mathcal{M}_l)_{\pi_l} \end{aligned} \quad (3)$$

Equation 3 is used as a heuristic for comparison more than as a true approximation of value. The weighed sum emphasizes more valuable states from likelier actions over other state-action pairs in $\mathcal{M}$. It does not evaluate actions not taken by the extant policy.

It should be noted that the controller only has access to $\mathcal{M}$ under π. There is not enough time to deploy each policy in the library and collect a buffer of experiences. Therefore, for equation 3 to hold, the ordinal relationships use the substitution $V(\mathcal{M})_{\pi_l} \approx V(\mathcal{M}_l)_{\pi_l}$ for $l \in L$. Figure 1 demonstrates how policies trained on tasks similar to M_t will share intermediate or terminal states. Therefore the valuation of the π_l using $\mathcal{M}$ will be truer to the valuation had $\mathcal{M}_l$ been used. With a truer policy evaluation, the policy likely to perform better will be chosen. The following experiments bear these results by showing the utility of policy re-use and its limitations under faults of various similarities.

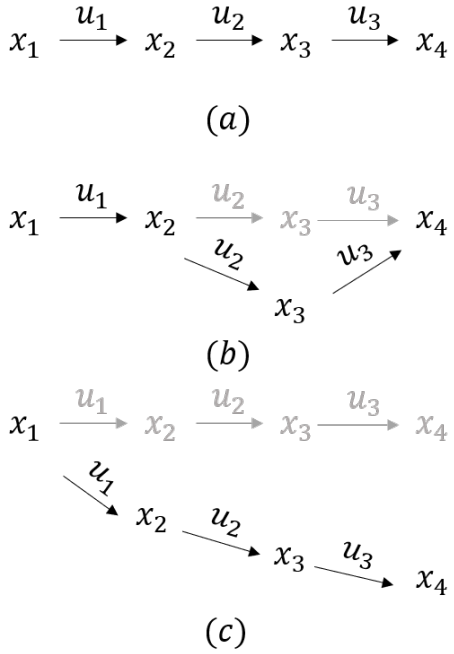

Figure 1: Illustration of buffered experiences using different policies under M_t. Dark sections represent stored experiences. Faint sections show the difference with the extant policy π under M_t. (a) : $\mathcal{M}$ using π. (b) : $\mathcal{M}_l$ where the task to train π_l is similar to M_t. (c) : $\mathcal{M}_l$ but where the task to train π_l is dissimilar to M_t.

3.3 Test-bed for experiments

The RL test-bed is a six-tank fuel transfer system on the wings of an aircraft (figure 2). Fuel is pumped from tanks to engines under a fixed schedule. The objective is to transfer fuel between tanks to keep fuel mass balanced about the longitudinal axis, to keep fuel mass concentrated at the extremities, and to conserve fuel mass against leaks. This is a hybrid system. The state space $x \in [0,1]^6$ constitutes of fuel levels in each tank as a fraction of their height. The action space is the status of valves on each tank $x \in \{[0..1]\}^6$. The environment is parametrized by the tank geometry, valve resistances, and engine fuel consumption rates.

The model can experience multi-modal faults. Each faulty state represents a separate task to be learned by the controller.

- Valve faults leave a single or a pair of valves unable to close fully.
- Leak faults cause fuel to leave tanks besides through engine pumps or valves.
- Pump faults cause fuel supply to engines degrade for a tank.

The evaluation metric for training and testing is the reward function. It is made up of several sub-components,

each pertaining to operating features that should be optimized:

- **centre** is the deviation of centre of gravity from the longitudinal axis. Closer to zero is better.
- **activity** is the average number of valves that are open. Closer to zero is better to prevent unnecessary mass transfer.
- **spread** is the variance of mass distribution about the longitudinal axis. Higher is better for forward-swept wings.
- **level** is the average fuel in tanks. Higher is better.
- **deficit** is the engine fuel demand unmet my pumps in tanks. Closer to zero is better.

Ultimately, the components form the reward function in equation 4. Figure 3 charts the reward components for each time step in a nominal episode, and the total reward as a function of those components.

$$r = \left(level + (1 - |centre|) + \frac{spread}{2} - \frac{activity}{4} \right) \cdot (1 - deficit) \tag{4}$$

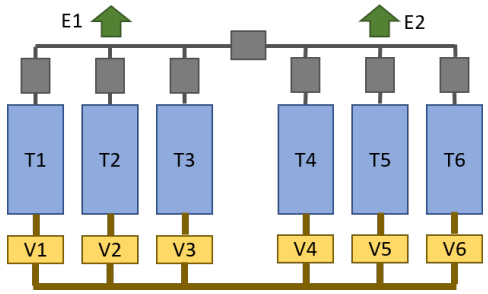

Figure 2: The fuel tank system. Pumps are in grey, tanks in blue, and valves in yellow.

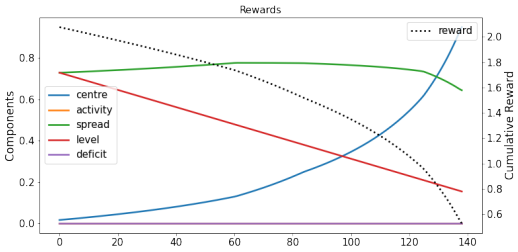

Figure 3: A breakdown of various components in the reward signal over a single episode. For the nominal case, components such as `deficit` and `activity` are zero.

4 Results

For the following experiments, a library of policies trained on faults was learned. In all cases, a fault was chosen without replacement and introduced into the environment. This created a new MDP representing the target task M_t. A proximal policy was chosen from the policies corresponding to the remaining faults, corresponding to the source tasks L. The proposed approach was evaluated using the area under curve (AuC) of reward signals over multiple episodes of operation after a fault. AuC was calculated for indivitual reward components, and the total reward function as described in equation 4. As a benchmark, the performance of the same policy before the fault was charted as the "Vanilla" approach, along with the averaged performance of choosing every other policy in the library, labelled "Average".

Type	Number	Description
valves25	4	Pair of valves stuck at 25%
valves50	4	Pair of valves stuck at 50%
leak	4	Leak in one fuel tank
pumps	4	Pump stuck at 10%

Table 1: A description of the designation and sizes of various modes of faults.

4.1 Measuring proximal policies

A library of 16 faults was used to measure the pairwise similarities between their corresponding policies. Table 1 shows the various modes of faults in the system. The Jensen-Shannon divergence was compared against the cosine, Euclidean, and L1 norm metrics. Figure 4 shows normalized pairwise distances between policies trained on faults for each similarity measure. Of note are the lower mutual distances for faults pertaining to valves (indices 1-8) and leaks (indices 9-12). They are delineated against pump faults (indices 13-16). One explanation for this is that the fluid dynamics of valves and leaks are similar: the flow rates depend on tank pressure and resistances. Whereas the flow rates for pump faults are modeled as constant. Figure 5 shows an alternative visualization where the distances are projected in a possible arrangement in two dimensions. By all similarity measures, the overlap between valve and leak policies is larger than with pump policies. To note is that all four similarity measures qualitatively delineate different fault modes. Whereas the use of Jensen-Shannon divergence was motivated in this work, its marginal utility over other measures demands further study.

4.2 Single-mode faults

The approach was first tested by having only a single category of fault in the library and occur in the system. In this and the following subsections, the domain of faults were sampled from [`valves50, leaks, pumps`] as described in table 1. Results are presented in table 2. For the valve and leak faults, the proposed approach produces higher overall rewards. This translates into an aerodynamically preferable mass distribution and lesser number of valves open. For pump faults, our approach is the least rewarding. From figure 4 and 5, pump faults have one of the larger mutual distances inside their category and against other categories of faults. This means that pump faults' policies are significantly different from each other, and therefore the estimate in section 3.2 may have been inaccurate.

4.3 Novel Faults

To evaluate the case where the encountered fault was completely novel or the learned policies were very different, the set of faults on the system and learned policies in the library were kept disjoint. Under novel faults, the performance was

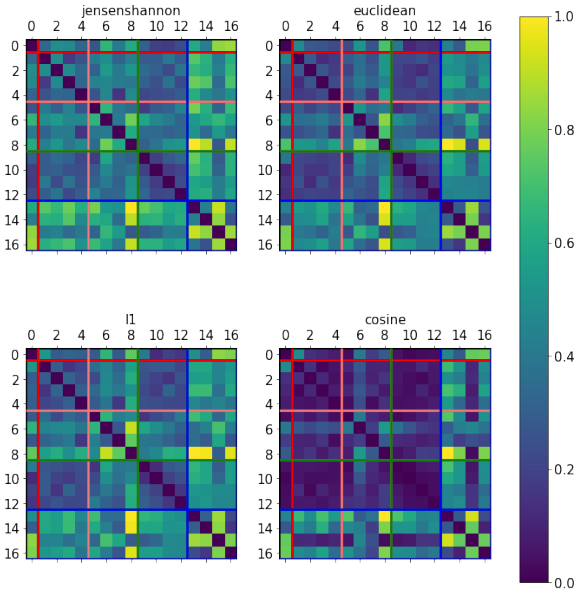

Figure 4: Pair-wise distances between the nominal policy (index=0) and faults (1-27) normalized to $[0, 1]$. The demarcating lines represent different fault modes as described in table 1.

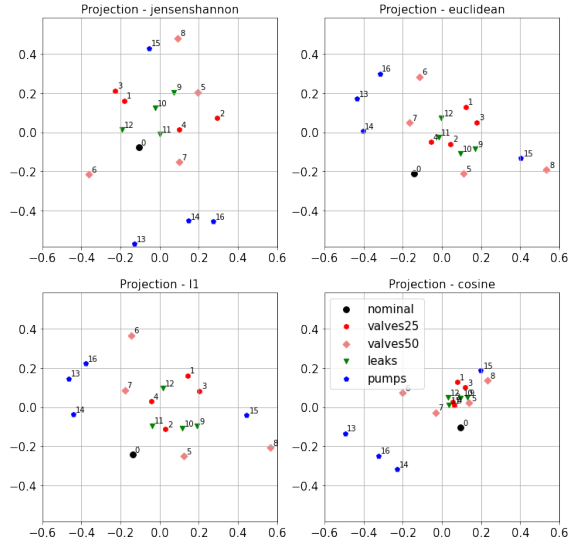

Figure 5: Using multi-dimensional sampling to project policies on 2 principal axes.

valves50	AuC					
	Deficit	Centre	Activity	Spread	Level	Total
Ours	**0.0325**	**262**	**28918**	**32158**	**17948**	**65735**
Vanilla	0.0432	367	29794	31867	17945	65298
Average	-	-	-	-	-	64598

leaks	AuC					
	Deficit	Centre	Activity	Spread	Level	Total
Ours	52.39	**-5.37**	**28512**	**31424**	**18013**	**64847**
Vanilla	**39.86**	-15.51	30161	31139	18008	64583
Average	-	-	-	-	-	64568

pumps	AuC					
	Deficit	Centre	Activity	Spread	Level	Total
Ours	1012	**-18.2**	32342	29429	17634	61724
Vanilla	**947**	-41.4	**29557**	**29487**	**17643**	62507
Average	-	-	-	-	-	**64568**

Table 2: Rewards from single-mode faults where library and encountered faults belong to a single category.

poorer. This reinforces the hypothesis in section 3.2 that when the optimal sequence of actions and states for a new task is significantly different from the buffer of experiences generated by a sub-optimal policy, the corresponding evaluation may not be useful for ranking.

valves50	AuC					
	Deficit	Centre	Activity	Spread	Level	Total
Ours	0.073	-35.5	**28572**	**32181**	**17946**	**65919**
Vanilla	**0.049**	**1.56**	29139	31858	17940	65563
Average	-	-	-	-	-	64686

leaks	AuC					
	Deficit	Centre	Activity	Spread	Level	Total
Ours	55.78	1111	33874.25	**30892**	**18008**	63394
Vanilla	**37.7**	**995.9**	**30181.75**	30812	18007	64398
Average	-	-	-	-	-	**64460**

pumps	AuC					
	Deficit	Centre	Activity	Spread	Level	Total
Ours	203	803	30403	**30485**	17915	63860
Vanilla	**171**	**763**	**28168**	30472	**17916**	**64459**
Average	-	-	-	-	-	64030

Table 3: Rewards from novel faults where library and encountered faults belong to a separate categories. Names are categories of the novel fault.

5 Conclusion

In this paper we presented a use case of transfer learning for fault-tolerant control of a hybrid system. By exploiting experiences from *similar* faults, the controller was able to achieve marginally better performance than using its current policy for learning the new task. In cases where the encountered fault was novel, or where the category of that fault resulted in *dissimilar* policies, the achieved performance often lagged behind simply learning using the extant policy.

Thus, the choice of a dynamic policy ranking metric which is robust against dissimilar policies poses an interesting problem for future work. Equally vital is the pruning of the library of policies to keep the sample size small

while preserving breadth of experiences. This paper presented some preliminary solutions by way of similarity measures to discriminate policies. Further study is required in the selection criteria for policy reuse, incorporation of meta-learning, and choice of hyper-parameters and regularization techniques against multiple test-beds to validate this approach thoroughly.

References

[1] Ibrahim Ahmed, Marcos Quiñones-Grueiro, and Gautam Biswas. "Fault-Tolerant Control of Degrading Systems with On-Policy Reinforcement Learning". In: *IFAC-PapersOnLine*. 21st IFAC World Congress. IFAC. 2020.

[2] Leemon Baird. "Residual algorithms: Reinforcement learning with function approximation". In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 30–37.

[3] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[4] Mogens Blanke et al. *Diagnosis and fault-tolerant control*. Vol. 2. Springer, 2006.

[5] Justin A Boyan and Andrew W Moore. "Generalization in reinforcement learning: Safely approximating the value function". In: *Advances in neural information processing systems*. 1995, pp. 369–376.

[6] H. Chen et al. "Data-Driven Fault Detection for Dynamic Systems With Performance Degradation: A Unified Transfer Learning Framework". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–12. DOI: 10.1109/TIM.2020.3033943.

[7] Ido Dagan, Lillian Lee, and Fernando Pereira. "Similarity-Based Methods for Word Sense Disambiguation". In: *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*. 1997, pp. 56–63.

[8] F. Fei et al. "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)*. 2020, pp. 7358–7364. DOI: 10.1109/ICRA40945.2020.9196611.

[9] WANG Hongm et al. "Data driven fault diagnosis and fault tolerant control: some advances and possible new directions". In: *Acta Automatica Sinica* 35.6 (2009), pp. 739–747.

[10] John MacGregor and Ali Cinar. "Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods". In: *Computers & Chemical Engineering* 47 (2012), pp. 111–120.

[11] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[12] Rui Nian, Jinfeng Liu, and Biao Huang. "A review On reinforcement learning: Introduction and applications in industrial process control". In: *Computers and Chemical Engineering* 139 (2020), p. 106886.

[13] W. Okamoto and K. Kawamoto. "Reinforcement Learning with Randomized Physical Parameters for Fault-Tolerant Robots". In: *2020 Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems (SCIS-ISIS)*. 2020, pp. 1–4. DOI: 10.1109/SCISISIS50064.2020.9322775.

[14] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[15] Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation." In: vol. 99. 1999, pp. 1057–1063.

[16] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[17] D. Zhang and Z. Gao. "Fault Tolerant Control Using Reinforcement Learning and Particle Swarm Optimization". In: *IEEE Access* 8 (2020), pp. 168802–168811. DOI: 10.1109/ACCESS.2020.3022893.

[18] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. "Transfer Learning in Deep Reinforcement Learning : A Survey". In: (), pp. 1–22. arXiv: arXiv:2009.07888v2.

[19] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. "Transfer learning in deep reinforcement learning: A survey". In: *arXiv preprint arXiv:2009.07888* (2020).