ENHANCED SAFETY THROUGH AI: AN ASSESSMENT OF ADVERSARIAL REINFORCEMENT LEARNING FOR CONTROL OF COMPLEX AIRCRAFT

Cynthia Koopman[1] & David Zammit-Mangion[1,2]

[1]University of Malta, Msida MSD 2080, Malta
[2]Cranfield University, Cranfield, Bedford, MK43 0AL, UK

Abstract

The integration of AI into commercial aircraft cockpits will constitute a significant leap forward in aviation technology but it prompts questions about necessity, benefits, and safety. This paper investigates a novel approach, employing Reinforcement Learning and introducing an adversarial agent, a 'Gremlin,' to manipulate aircraft systems strategically and attempt to deceive an 'Angel'— the Reinforcement Learning agent responsible for control. In combination with a large task space and a non-restrictive reward function, this approach aims at creating a deep understanding of the systems on the aircraft and its surrounding in the AI system to ensure correct response in unforeseen conditions. System failures and upset conditions are induced to strengthen the knowledge of the AI system. Through rigorous testing, we aim to provide insights into the robustness and limitations of the system in unforeseen circumstances, contributing to the advancement of safe AI integration in aircraft cockpits.

Keywords: AI, robust control, upset recovery, Reinforcement Learning

1. Introduction

In the evolving landscape of aviation, the integration of artificial intelligence (AI) into the cockpit of commercial aircraft is a large step forward. This raises a number of questions, such as whether it is necessary, what the benefits are, and how to create a comprehensive approach to ensure safety and efficacy. At the core of these questions is the recognition that AI has unique advantages, including the potential for task execution that surpasses human capabilities, presenting a compelling case for its integration into the cockpit.

An AI system capable of processing information from all aircraft systems has the potential of generating valuable insights, leading to improved safety. As greater levels of automation are being introduced in the cockpit, AI can be used to mitigate the persistent gap in human-machine interaction. By taking into account the limitations of humans and leveraging the strengths of machine intelligence, safety measures can be strengthened. An AI system, capable of identifying system failures and understanding their consequences, emerges as a critical component in contributing to overall safety improvement.

For safe integration of AI into the cockpit, one aspect that is crucial is to make AI responses predictable, especially in unforeseen circumstances. The real-world application of controlling an aircraft, with the potential for inadvertent pilot actions, system failures, and upsets, requires AI systems to not only perform well but also do so dependably even when faced with abnormalities.

A comprehensive strategy for creating such an AI system involves developing an understanding of the interaction of the onboard systems with the aircraft and its surroundings, one that is deep enough to correctly respond in conditions that may not have been encountered previously. Creating such an understanding in AI involves maximizing exploration, automating the generation of training tasks with appropriate difficulty levels, and leveraging transfer learning abilities. By allowing an AI system to

explore and develop their capabilities to the fullest extent, a greater potential to adapt to a multitude of complex situations is unlocked.

This paper investigates the integration of AI into commercial aircraft cockpits, exploring the benefits and limitations of a novel, comprehensive approach to creating an AI system aimed at enhancing safety in unforeseen circumstances for aircraft control. For this approach, an AI system based on Reinforcement Learning was trained using an adversary (the 'Gremlin') to identify weaknesses during its training phase. The adversary is responsible for introducing system failures and inducing upset conditions. By subjecting the resulting Reinforcement Learning system to rigorous testing, we aim to provide insights into the robustness and limitations of the system in unforeseen circumstances.

The remainder of this paper is structured as follows. Section 2. presents the concept of the novel competition-based training method to create the AI system. Section 3. provides details of the implementation. Section 4. discusses the evaluation metrics and tests for which the results are then presented in Sections 4.1 - 4.3.

2. Competition-based training: Angels and Gremlins

This paper tests a system in which artificial neural networks were trained using Reinforcement Learning to develop an AI system capable of generating control commands. These commands can be used by the pilot or temporarily assumed by the AI to control the aircraft. More importantly, Reinforcement Learning was used with the aim of maximizing the ability to learn about the aircraft in its environment, as well as the effect of interactions. Given the challenging nature of identifying optimal actions in various situations for this application, Reinforcement Learning emerges as a compelling approach as it also eliminates the need to define optimal outcomes or labels. In Reinforcement Learning an agent repeatedly acts in an environment based on current state information about the environment. Collected state-action pairs together with a calculated reward for each action form a dataset from which the neural network of the agent can be optimized. The concept of having an AI system understand the systems on the aircraft with Reinforcement Learning has been investigated in previous work. In [1] this resulted in superior performance when faced with reconstruction of Pitot-static failures, and in [2] the Reinforcement Learning agent was able to combine multiple flying tasks in a non-sequential manner.

To train a Reinforcement Learning agent in a comprehensive manner, it is important that the agent is exposed to enough examples during training from which meaningful features can be formed. As the Reinforcement Learning agent learns from its own interactions with the environment, exposing the agent to such examples is not straightforward. In this paper, we test a competition-based training method to provide the Reinforcement Learning agent with examples that lead to an effective training schedule, which extends the abilities of the Reinforcement Learning agent previously investigated in [1] and [2].

In the proposed methodology, an adversary, referred to as the 'Gremlin,' is introduced as a Reinforcement Learning agent. This Gremlin engages in a competitive dynamic with the Reinforcement Learning agent responsible for aircraft control, referred to as the 'Angel.' The historical origin of the term 'Gremlin' traces back to its attribution by RAF pilots during World War II, where it was ascribed to malfunctions in aircraft. In alignment with this historical context, our approach utilizes the Gremlin to manipulate aircraft systems strategically, aiming to deceive the Angel.

Within the field of AI, multi-agent competition using Reinforcement Learning has previously demonstrated unique advantages [3] [4], which can be useful in developing an AI system for aircraft control with safety and comprehensiveness in mind. Within this context, the Gremlin could force the Angel to confront and address its weaknesses, yielding specific advantages. Coupled with a large task space and an non-restrictive reward function, the Gremlin would facilitate extensive exploration by the Angel. This exploration could mitigate suboptimal convergence, allowing the Angel to delve into crucial subsets of the environment, of which it might not have been motivated to do so before. As both Angel and Gremlin dynamically adjust their parameters during training, they evolve and maintain competitiveness, establishing an autocurriculum that automates training task difficulty levels for the Angel.

The introduction of the Gremlin creates significant variability in tasks that may previously be considered mundane. This variability, coupled with the large training space and a non-restrictive reward

function, can enhance the system's transfer learning capabilities. In this way, the Angel can be incentivized to develop a deep understanding of environmental dynamics rather than relying on superficial features, thereby enabling it to respond correctly in unforeseen conditions.

3. Implementation details

3.1 Reinforcement Learning architecture

The on-policy Proximal Policy Optimization algorithm [5] was used to implement Reinforcement Learning. This algorithm trains two neural networks, an actor and a critic. The actor network provides the actions and the critic provides the actor network with an estimate on the future reward based on the current state. The critic network is optimized using the future discounted reward and the actor network is optimized based on the advantage of taking the estimated actions. Both actor and critic networks were implemented having an architecture of an LSTM layer followed by two fully connected layers which all had 256 neurons. The parematers were not shared between the neural networks of the actor and critic.

Training of these neural networks was performed by performing tasks in the JSBSim [6] simulation software using the A320 commercial aircraft model. The output of the actor network in the Angel agent were the actions representing the continuous control of elevator deflection and thrust setting. Spins were not considered due to simulation limitations. The output of the actor network in the Gremlin agent was a manipulation of the angle of attack sensor with an offset between -10 and 10 degrees. Such a manipulation of the angle attack sensor can challenge the dependencies and understanding of the Angel as it has to depend less on the angle of attack input and more on the related relationships of other variables to understand if the angle of attack information is useful.

The inputs, or observations, to the Reinforcement Learning agents (both Angel and Gremlin) were the inputs and outputs of the systems available during flight. These included measurements of altitude, airspeeds, attitude, pressure, temperature, speed of sound, weight and current positions of controls. Information about the requested airspeed and altitude targets was also included, allowing the agent to create relationships between the reward and the task to be solved. The Angel and Gremlin received the same states as input, except that the states seen by the Angel may be manipulated by the Gremlin.

3.2 Training task space

The tasks (episodes) that were performed by the Angel agent during training were randomized in terms of initial position, weight, and target altitude and airspeed. To ensure possible combinations of initial values, the episode only started when the aircraft was trimmable using the built-in function provided by JSBSim. This was done to limit overfitting to an episode that always starts in an upset or abnormal condition. The initial altitudes were sampled between 15,000 and 35,000 feet and the initial airspeeds between 420 and 770 ft/s true airspeed. The weight of aircraft was varied altering the fuel load and was sampled between a total aircraft weight of 52 and 72 tonnes. The target altitude and airspeed were also randomly sampled at the beginning of the episode; altitude change was sampled between -1000 and 1000 feet, and calibrated airspeed change between -100 and 100 ft/s. Restrictions to the training space were only included to conform with the simulation limitations. Therefore, the episode was prematurely ended when exceeding the maximum operating speed or mach (Vmo or Mmo) to match the design limits of the aircraft; the altitude was lower than 500 ft, to avoid empty values from the simulation; and the angle of attack exceeded 30 degrees, as this was not modeled. Prematurely ending the episode has a strong influence on the training of the agent as it potentially loses a large amount of reward and therefore learns to avoid such states.

3.3 Reward function

In the randomly sampled tasks during training, the responses of the Angel and Gremlin are evaluated by the reward function. This reward is used to optimize both networks to enforce the desired behavior. The reward provides feedback to the Reinforcement Learning agent of its actions in the environment. Not only does this reward guide the agent towards solving the tasks, it also indirectly affects the training space that the agent will see in the future optimization iterations. In the proposed approach, a reward function that is generalizable to any task in any area in the environment was implemented.

For the Angel, the reward function focused mainly on the distance to the altitude and airspeed targets set at the beginning of the episode. This target reward is summarized in Eq. 1:

$$r_{target} = sigmoid(r_{\Delta_{altitude}}) + sigmoid(r_{\Delta_{CAS}}), \tag{1}$$

where the sigmoid function is $sigmoid(x) = \frac{1}{1+e^{-x}}$. The sigmoid function not only normalizes the reward but also has the effect that far from the target large improvement can be made by the Reinforcement Learning agent when moving in the correct direction, and close to the target the agent has to be more precise to gain more reward.
In addition, a small reward was included to stay within vertical speed, vertical acceleration, and angle of attack limits. Although the Angel would theoretically receive enough feedback from the environment to not prefer to go into an aerodynamic stall, it was found from early experiments that, at least within JSBSim, the Reinforcement Learning agent was able to exploit regions by for example staying just below the critical angle of attack. Therefore, a small reward was added when the angle of attack continued to decrease below 10 degrees. These reward components are grouped together and follow in Eq 2:

$$r_{limits} = r_{\dot{h}} + r_{a_y} + r_{aoa}, \tag{2}$$

where all components are 1 when within defined limits and 0 otherwise.
It was also experimentally found that very fast oscillations and erratic behavior had little effect on the elements included in the reward function. Therefore, it was beneficial to include a small reward which discourages large changes between actions in the previous and current step, following in Eq 3:

$$r_{actions} = r_{\Delta_{elevator}} + r_{\Delta_{thrust}}. \tag{3}$$

The complete reward function for the Angels then follows as in Eq. 4:

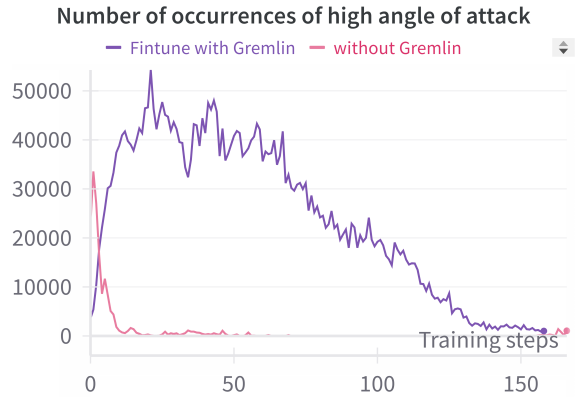
$$R = \frac{1}{N} \sum w_1 r_{target} + w_2 r_{limits} + w_3 r_{actions} \tag{4}$$

where N is the number of reward elements that are included in the reward function which are weighted by w_1, w_2, and w_3.
For the Gremlin, a sparse reward was given when the angle of attack exceeded 15 degrees and the lift coefficient decreased (stall). This was implemented in combination with the Gremlin's ability to manipulate the angle of attack sensor with the aim to deceive the Angel and force an aerodynamic stall.

3.4 Training

Two training runs were performed, with and without the Gremlin. The Gremlin was used in a fine-tuning setting where the training started with an Angel that was previously trained for 80 training steps on its own. It was found that such pre-training can allow the Angel to first learn basic skills. Training graphs are shown in Fig. 1. Figure 1a presents the mean reward obtained during training, which is similar for both with and without the Gremlin. The influence of the Gremlin becomes visible in Fig. 1b, where the total occurrences of high angle of attacks are summarized. Without the Gremlin, the Angel quickly reduces the number of occurrences of high angles of attack to mean zero. With the Gremlin, the Angel clearly faces new challenges (it started pre-trained from 80 training steps) and needs around 150 steps to win over from the Gremlin. A successful convergence towards learning to fly whilst experiencing angle of attack sensor failures is indicated by the resulting reward being as high as without the Gremlin (Fig. 1a). The following sections will discuss and present the results to test this ability, as well as the knowledge gained by Angel to respond in unforeseen conditions.

(a) Average reward of the Angel.



(b) Total occurrences of angles of attack exceeding 15 degrees.

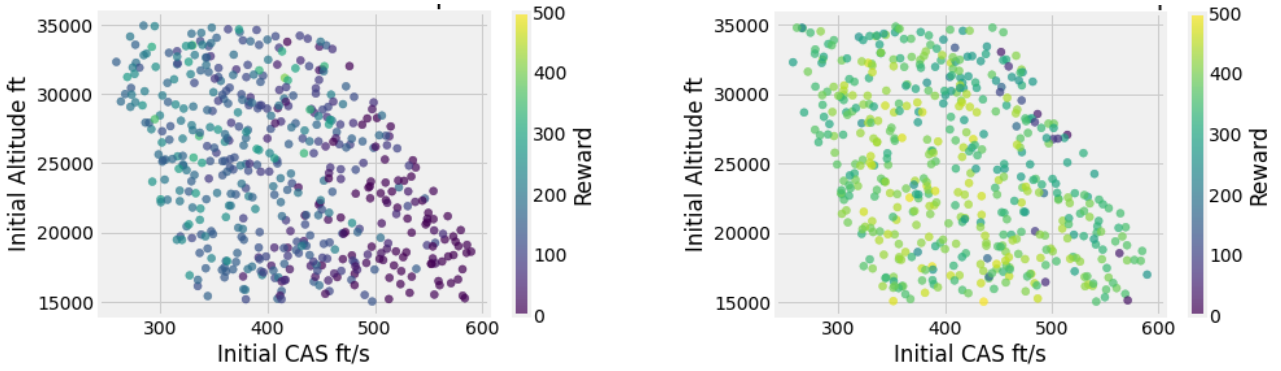
Figure 1 – Training graphs showing the reward of the Angel and the total number of occurrences of angles of attack exceeding 15 degrees for training with and without Gremlin. Every training step consisted of 256 episodes each with 1200 steps, resulting in one training step optimizing over 1200 * 256 episode steps taken by the Angel and Gremlin.

4. Testing the Reinforcement Learning agent

The resulting Angel, which was fine-tuned using a Gremlin, was tested under various conditions. The Angel trained without Gremlin was used as a baseline. The reward function of the Angel served as one evaluation metric as it is not only what the Angel directly optimizes for, but also combines factors that are informative for the performance. An episode has a length of 1200 steps at 10 Hz (120 seconds) and receives a reward at every step. The maximum reward is in practise far less than 1200 as, for example, the initial states are rarely the target states and the time required to go to targets is limited by factors such as the vertical speed reward. Experimentally, a reward above 300 can be considered successful. Tests were performed to assess the operational boundries of the resulting Angel including tests in unforeseen circumstances where the focus was on stall recovery and shift in center of gravity. As the tasks of the Angel were formulated as combinations of target airspeed and target altitude, the stall recovery manouvre in these tests was extended to also include going to a specified altitude and airspeed. To evaluate these stall recoveries, various metrics were used in addition to the reward; maximum altitude loss, time until the angle of attack was reduced below 15 degrees, and the ability to return to the target altitude and airspeed. For the stall recovery tests, the altitude target was to return to the initial altitude and the target airspeed was to increase by 70 ft/s ($\sim$ 41 knots). All stall recovery tests were performed at different weights and heights; total aircraft weight was sampled between 60 and 70 tonnes, and the stall was induced at either 20,000, 25,000, 30,000, or 32,000 ft.

4.1 Analyzing operating boundaries with a malfunctioning angle of attack sensor

The operating limits as well as the robustness of the proposed method can be analyzed by looking at the performance across different combinations of altitudes and airspeeds. This is shown in Fig. 2 for different trimmable combinations of initial altitude and airspeed while random manipulations affected the angle of attack sensor. The scatter plots in Figs 2a and 2b show the rewards obtained with the angle of attack failure for training without and with a Gremlin manipulating the angle of attack sensor during the training phase. The test shows that with the proposed method, the Angel is able to generalize and thus achieve high reward for most initial conditions. Training with the Gremlin has also been shown to have a significant effect on the reward of the Angel and improves the robustness of the response of the Angel as the performance is superior to that without Gremlin.



(a) Training without the Gremlin.

(b) Training with the Gremlin.

Figure 2 – Episodic rewards obtained with Angel trained with and without the Gremlin for tasks at 500 randomly chosen initial altitude and airspeed combinations, with random angle of attack sensor offsets between -10 and 10 degrees.

4.2 Stall recovery with a malfunctioning angle of attack sensor

The proposed method trains the Angel only on tasks that are simple combinations of target altitude and airspeed starting from a trimmed position. Due to the randomized nature of Reinforcement Learning in early stages of training as well as the lack of limitations included during training, scenarios that resemble aerodynamic stalls are not excluded. In fact, the proposed method intentionally does not restrict going close to or into a stall during training, except for a small discouragement in the reward function. The feedback of the environment that the agent will receive by going into a stall will itself be disadvantageous to the agent, as it would experience control which is not optimal to maximize reward. Therefore, the agent will indirectly learn not to approach a stall and get out of it quickly. Other training approaches may, via more direct reward components, achieve stall prevention and recovery with less training time and examples. However, by not introducing any reward explicitly directed at stall, our approach does not risk the agent learning shallow features based on the reward. Instead, it creates an environment where the agent has to learn the dynamics and the implications of stall. It thus develops much deeper features which may also be relevant in other situations, enabling transfer learning abilities that facilitate multi-task learning. This is considered advantageous.

The ability to recover from a stall was tested by manually creating stall scenarios at different altitudes which resulted in stall for weights between 60 and 70 tonnes. The Angel trained with and without Gremlin was tested on stall recovery in four different angle of attack failure settings and a nominal setting without failure. Table 1 and Table 2 show several metrics over 100 random runs for both conditions, respectively. When training without the Gremlin, the performance seems not to be heavily affected by random manipulations. However, examples show that the actions become very erratic resulting in large and fast angle of attack deviations. The Angel trained without the Gremlin is affected the most by a continuous increase or decrease in angle of attack information. The performance when trained with the Gremlin, shown in Table 2, does not show such an effect. When trained with the Gremlin, the Angel is able to keep the performance constant when performing stall recovery.

An example comparing both Angels trained with and without the Gremlin with a decreasing angle of attack manipulation are shown in Figure 3 and Figure 4 respectively. Here, the response of the Angel without the Gremlin is clearly affected by the angle of attack manipulation, and the Angel is tricked into a secondary stall. However, when trained with the Gremlin, the stall is recovered from and the target altitude and airspeed are reached successfully with a high degree of precision, which indicates that the Angel was able to triangulate enough information without having correct angle of attack information available.

	Mean reward	Max alt loss [ft]	aoa>15 [sec]	Return alt [%]	Return CAS [%]
Random fixed	209.00	1644.00	0.48	16	87
Random	215.75	1540.25	0.68	18	90
Decrease	112.75	1551.25	35.16	16	0
Increase	111.75	1531.25	35.84	16	0
Nominal	277.50	1363.50	0.53	32	100

Table 1 – Stall recovery results for training without the Gremlin, showing averaged results for testing 100 episodes each at 20,000 ft, 25,000 ft, 30,000 ft, and 32,000 ft. The weight ranged between 60 and 70 tonnes. Return to altitude and calibrated airspeed (CAS) are percentage of episodes successfully within 200 ft of the target altitude and 20 ft/s ($\sim$ 12 knots) deviation of the target airspeed.

	Mean reward	Max alt loss [ft]	aoa>15 [sec]	Return alt [%]	Return CAS [%]
Random fixed	273.25	1356.50	9.92	76	89
Random	273.75	1390.50	10.47	75	91
Decrease	270.75	1409.50	10.15	77	93
Increase	271.25	1385.00	10.45	79	91
Nominal	272.25	1386.75	10.23	82	90

Table 2 – Stall recovery results after fine-tuning with the Gremlin, showing averaged results for testing 100 episodes each at 20,000 ft, 25,000 ft, 30,000 ft, and 32,000 ft. The weight ranged between 60 and 70 tonnes. Return to altitude and calibrated airspeed (CAS) are percentage of episodes successfully within 200 ft of the target altitude and 20 ft/s ($\sim$ 12 knots) deviation of the target airspeed.

4.3 Stall recovery with a shift in center of gravity

The center of gravity was changed to assess the Angel's response to unforeseen (untrained) conditions as well as to understand the knowledge it has gained between the response of the aircraft and the information it receives. The testing was again performed in the stall scenarios to create difficult examples of center of gravity not previously experieced by the Angel. Table 3 shows the average results for 100 random episodes with a center of gravity shift of 50 inches forwards and backwards. The results are shown for different altitudes at a fixed weight of 62 tonnes where the stall speed is approximately 185 knots CAS. As in the previous section, the Angel fine-tuned with the Gremlin was not affected by angle of attack manipulations and therefore only results without angle of attack manipulations are shown. These results show that the change in center of gravity has the largest effect in maximum altitude loss and the return to initial altitude.

Altitude	Mean reward	Max alt loss [ft]	aoa>15 [sec]	Return alt [%]	Return CAS [%]
20000 ft	268.00	838.00	3.35	30	100
25000 ft	251.00	1047.50	6.79	35	100
30000 ft	220.50	1301.00	7.74	61	100
32000 ft	176.50	1090.00	13.93	51	47

Table 3 – Stall recovery results after fine-tuning with the Gremlin, showing averaged results for testing 100 episodes at different heights with the center of gravity either 50 inches forwards or backwards. The weight ranged between 60 and 70 tonnes. Return to altitude and calibrated airspeed (CAS) are percentage of episodes successfully within 200 ft of the target altitude and 20 ft/s ($\sim$ 12 knots) deviation of the target airspeed.

An example is shown in Figure 5, where the center of gravity is shifted between all 3 positions (center, forward and aft). An aft center of gravity resulted in an initial oscillation and, overall, a reduced ability to recover the altitude. The relative challenge in recovering from stall with an aft center of gravity is related to the flight dynamics. A rear center of gravity reduces the longitudinal stability, reduces

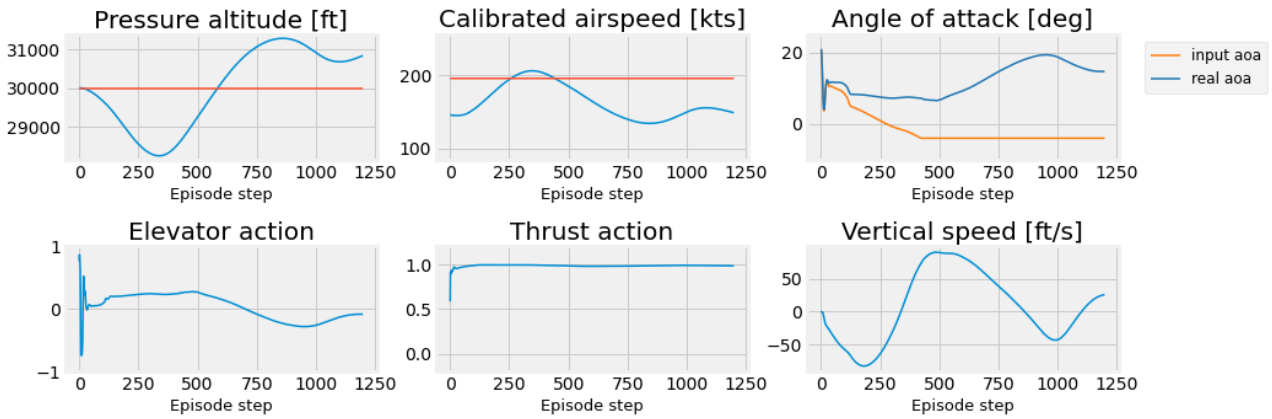


Figure 3 – Stall recovery at 30,000 ft at a weight of 62 tonnes with the Angel trained without the Gremlin.

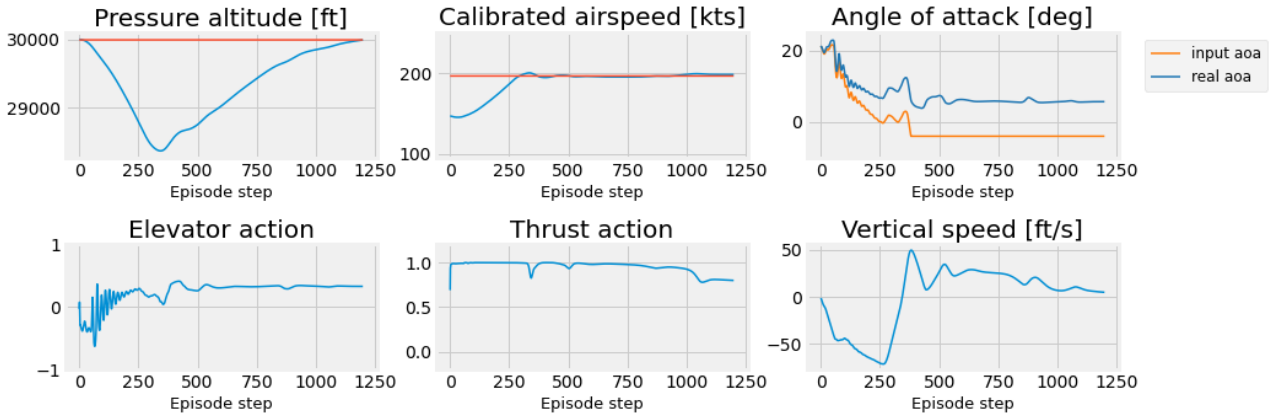


Figure 4 – Stall recovery at 30,000 ft at a weight of 62 tonnes with the Angel fine-tuned with the Gremlin.

the stiffness of the aircraft in pitch, and increases the pitch down moment required of the horizontal tail. All in all, this delays stall recovery. More importantly, the Angel showed that it can recognize the change in the behavior of the aircraft and is able to fly at such different configurations to recover and increase speed in a precise manner.

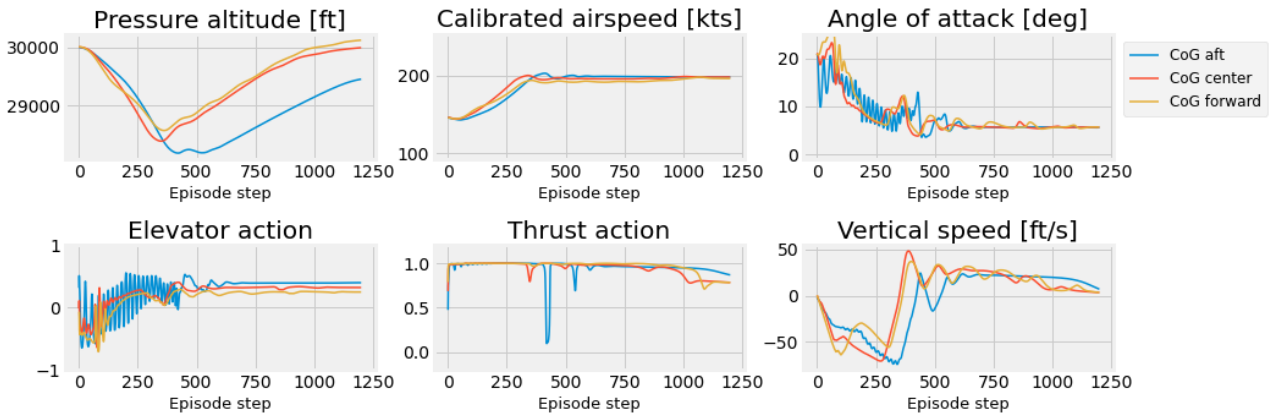


Figure 5 – Stall recovery at 30,000 ft and 62 tonnes for three different center of gravity (CoG) configurations; forward, center, and aft.

5. Conclusion

In this paper a novel method for training an AI system to understand and control an aircraft was tested. The competition-based method that was trained using neural networks and Reinforcement Learning had the aim of automating task difficulty during training by targeting weaknesses of the trained system. The results showed that when the system was trained with this method, it was able to successfully control the aircraft under angle of attack manipulations throughout the training range of altitude and airspeed combinations. This capability not only demonstrates robustness to angle of attack malfunctions but also that the Reinforcement Learning agent comprehends the environmental variations and the effects of the controls. The results of stall recovery tests showed that, by training on combinations of altitude and airspeed, indeed a more complex behavior was created. Not only was the AI system able to recover from stalls at various weights and altitudes, it was able to do so with a malfunctioning angle of attack sensor. To further increase the difficulty level, the stall recovery was also tested with changes in the center of gravity. These results showed that the proposed method has gained such a level of understanding of the aircraft and its environment that it was able to control the aircraft in conditions that were not only never seen during training but were also substantially more difficult. The performance demonstrated in these tests provided insight and showed promise for the proposed method to enhance safety by being able to react in a predictable way when faced with unforeseen circumstances.

6. Contact Author Email Address

The corresponding author, Cynthia Koopman can be contacted at the following email address: cynthia.koopman@um.edu.mt

7. Copyright Statement

References

[1] Cynthia Koopman and David Zammit-Mangion. Artificial intelligence for real-time tolerance to critical flight data errors in large aircraft. *Journal of Aerospace Information Systems*, pages 1–9, 2024.

[2] Cynthia Koopman and David Zammit-Mangion. Using reinforcement learning for ai systems in the mitigation of automation failures and stall recovery in complex aircraft. In *AIAA SCITECH 2024 Forum*, 2024.

[3] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *ICLR 2018*, 2018.

[4] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *ArXiv preprint*, 2020.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv preprint*, 2017.

[6] Jon Berndt. JSBSim: An open source flight dynamics model in C++. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 4923, 2004.