Learning To Fly

Andrews George Varghese
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, Maharashtra 400076
`180070005@iitb.ac.in`

Yash Vinesh Gadhia
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, Maharashtra 400076
`180100130@iitb.ac.in`

Abstract

Autonomous driving is fast becoming a reality today, particularly in the case of ground vehicles. There is naturally an increasing interest in autonomous flying machines, which are seen as the future of flight. However, conventional PID controllers are often insufficient to control large systems like the Airbus A320, particularly with changing setpoints, and more modern control approaches require highly accurate physical models and extensive parameter tuning in order to achieve even basic control. In this work, we present a policy search approach that uses a dynamic, piece-wise objective function to efficiently search the policy space of the aircraft control problem and show that it performs far better than a basic PID controller and even the PPO algorithm applied to an A320 simulated in a gym environment.

1 Introduction

Autonomous flying machines are the future of aerial vehicles and they present an area of great interest to the control engineering fraternity. Conventional autopilot systems are commonly implemented through complex nested proportional-integral-derivative (PID) controllers, which require tedious parameter tuning and are vulnerable to control instability in perturbed flight conditions. Previous works applying reinforcement learning to aircraft control have been limited to methods simplifying the problem by discretizing the action space and reducing the number of dimensions in the state space (Rennie [2018]). There is hence a need to explore more Artificial Intelligence based methods for autonomous flying which achieves full six degrees-of-freedom aircraft control with continuous actions and move towards a new generation of self-flying machines that operate safely and robustly.

Thus arises the problem we tackle in this work. The problem is part of the Airbus Learning to Fly Challenge (link). It deals with autonomous control of aircrafts. This problem domain is well suited for reinforcement learning which enables agents to learn from interactions with an environment. Also, reinforcement learning has been successfully used in the past for many control system problems including autonomous flying of helicopters (Ng et al. [2003]) and recent advances in the application of deep neural networks to RL have allowed agents to perform well in increasingly complex tasks, including continuous control tasks (Rennie [2018]).

2 Problem Setting

The task at hand is known as the `Heading Control Task` as provided in the `gym-jsbsim` environment. In this environment, we pilot an Airbus A320 trying to maintain a fixed altitude while achieving various target setpoints.

The task at hand The `Heading Control Task` requires our controller to be able to achieve the following conditions:

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

- Achieve stable, steady flight at fixed altitude 10000ft
- New target headings are provided every 150 seconds with increasing complexity. The complexity is increased by increasing the difference between the current and new target headings. i.e. difference between current and next headings increases linearly - ± 10 deg, ± 20 deg, ± 30 deg
- Termination of the task occurs when the current heading is not within 10 deg of the target heading at the end of 150 seconds, or if the aircraft is more than 100ft away from the target altitude
- Reward depends on deviations in heading and altitude throughout the duration of the flight

State Space The state space is a Nine-dimensional vector representing the altitude, heading, roll, pitch, yaw and velocity of the aircraft. Specifically, it is an array of [Delta Altitude, Delta Heading, Pitch, Roll, Downward Velocity, Total air speed, Roll rate, Pitch rate, Yaw rate]

Action Space We are given control over four actions, namely the Aileron (which controls roll), Elevator (which controls pitch), Rudder (which controls yaw) and the Throttle (which controls the aircraft's power).

3 Theoretical Background

We use various approaches to solve this problem, starting with a simple PID-based baseline, a PPO-based approach and ending with policy search and hill climbing.

PID Controller The PID controller (Fig. 1), or the Proportional-Integral-Differential controller, is a control loop mechanism employing feedback that is widely used in applications requiring continuously modulated control. The controller constantly calculates the error in the target property (i.e. the difference between the desired set-point $r(t)$ and the measured process variable $y(t)$) and uses three terms, namely a term proportional to the error, one which is the integral of the error, and the third being the derivative of the error, to produce the required correction ($u(t)$).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

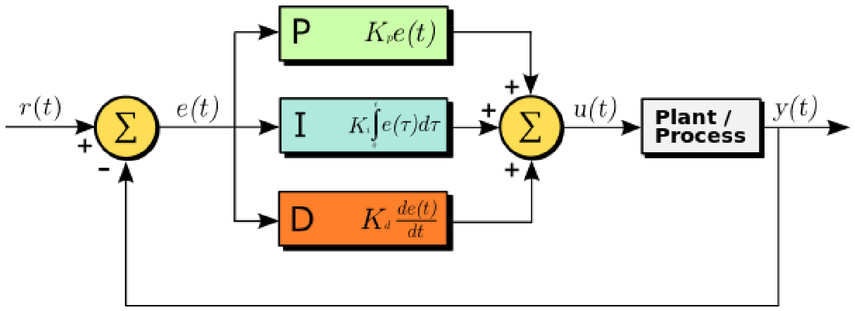


Figure 1: A block diagram of a PID controller in a feedback loop. $r(t)$ is the desired process value or setpoint (SP), and $y(t)$ is the measured process value (PV). Image taken from Wikipedia contributors [2022]

Proximal Policy Optimization PPO is a policy gradient method that improves on Trust Region Policy Optimisation using a clipped surrogate objective.

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta (a_t \mid s_t) \hat{A}_t]$$

$$r_t(\theta) = \frac{\pi_\theta (a_t \mid s_t)}{\pi_{\theta_{old}} (a_t \mid s_t)}$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t\right)\right]$$

Policy Search Policy Search is an application of Black Box Optimisation to Reinforcement Learning. The goal for black box optimisation is to use numerical derivative-free methods to optimize an objective function $f : W \to \mathbb{R}$ (where $W \subset \mathbb{R}^n$)

Since black box optimisation does not involve gradient computation, the advantage with using such numerical methods is that we need little to no assumptions about the function f i.e. it can be discontinuous, non-linear or erratic. The only requirement to running such methods is that it should be relatively efficient to compute f(w) (generally using computer simulations). In terms of the success of these methods, they are known to work well with problems that are in general more difficult to solve compared to convex optimization (Audet and Kokkolaras [2016]). However, most of these methods are heuristic based and do not have optimality guarantees but the ambition here is to find good parameter values efficiently rather than searching for optimal solutions.

In the context of reinforcement learning, the set W is the set of parameters defining our agent policy $\pi : S \to \mathbb{A}$ and $f(w)$ can be the expected long term reward that we wish to optimize. Random or Grid search over the set W are two natural approaches to solve this problem but they quickly fail with increase in the number of dimensions n. Also, no numerical method is expected to work well for a large number of dimensions (1000's or higher). Local search is known to work well for intermediate n (10's,100's). One common approach to local search is hill climbing which involves starting with an initial guess w_0 and then sampling points on a hypersphere in n dimensions with w_0 as the center. The idea is to compare $f(w_0)$ to $f(w)$ for all the sampled points and move to the point which gives the highest increase in the function value. Thereafter, it becomes the new center and the process repeats. The algorithm stops when the center gives the highest function value.

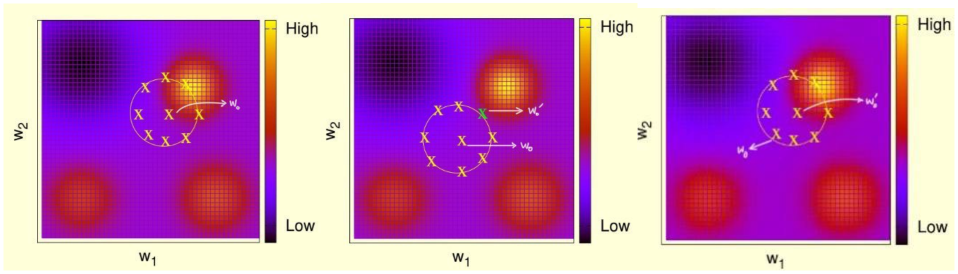


Figure 2: Illustration of hill climbing. The image is taken from: `https://www.cse.iitb.ac.in/shivaram/teaching/old/cs747-a2021/index.html`

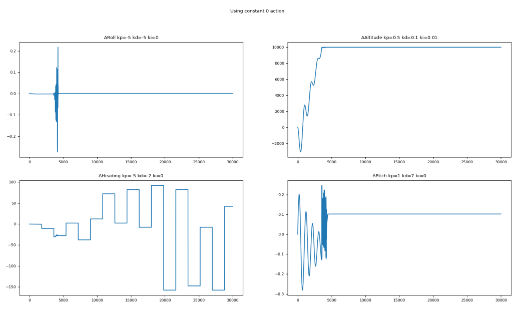
4 Experiments and Results

PID Controller In order to tune the PID controller, we broke the control problem into controlling altitude first and then the heading. We aimed to maintain roll and pitch while performing the above mentioned control tasks. We use the tuning strategy proposed by Ziegler and Nichols [1993] to tune the same.
While tuning for altitude control, no setting of the K_p, K_i, K_d values seemed to provide stable altitude flight over long time periods. We thus tried using an offset in the elevator and throttle actions, and were able to provide near-stable altitudes over long time periods. We then tuned the PID controller over these offset action values. We finally obtained a PID controller(PID constants are provided in Table 1) which was able to satisfactorily control roll, pitch, heading and altitude but was not able to control acceleration. These results (shown in Figure 3) corroborate findings that controls more complicated than PID are required to manoeuvre aircrafts.

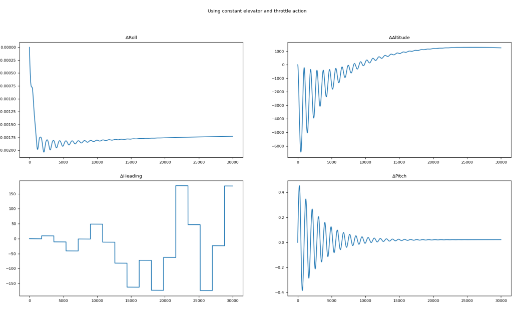
Proximal Policy Optimization In order to train agents using PPO, we first tried to learn a policy using the default reward function provided by the gym-jsbsim environment. The default reward function was a geometric mean of scaled gaussian rewards for each relevant variable i.e. heading, altitude, roll, speed and acceleration where the reward for each variable was

$$Reward = exp(-(VariableValue/ErrorScale)^2)$$

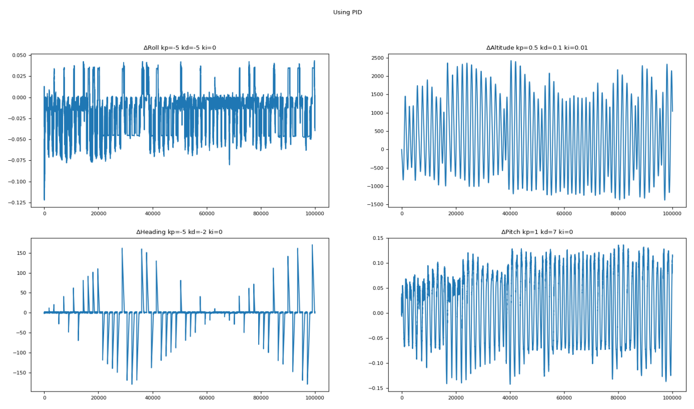
The error scales for the different variables are as in Table 2.



(a) Constant(0) action values



(b) Constant (non-zero) elevator and throttle action



(c) The baseline PID controller

Figure 3: PID controller

We trained PPO for 1M steps with default hyperparameters as given in StableBaselines3 (Raffin et al. [2021]).

Next we tried to modify the reward function by linearizing it to see if it improved performance. Thus instead of a geometric mean, we summed up rewards for each relevant variable, where the reward for each variable was -1 if it was beyond the error scale and a linear function between 0 to 1 if it was within the error scale. Finally each individual reward was given a weight and the weighted sum was the total reward. The weights for different reward terms are in Table 3.

We again trained PPO for 1M steps with the modified reward function and default hyperparameters. The results of the two PPO models are provided in Table 4

Table 1: PID constants for different actions

Action	K_p	K_i	K_d
Aileron	-5	0	-5
Elevator	1	0	7
Rudder	-5	0	-2
Elevator	0.5	0.01	0.1

Table 2: Error Scales

Variable	Error Scale
Heading	5 degrees
Altitude	50 feet
Roll	0.35 radians
Speed	16 fps
Acceleration(x, y)	0.1g
Acceleration(z)	0.5g

Policy Search For applying policy search to this problem, we modelled our policy $\pi : S \to \mathbb{A}$ as a neural network with a single hidden layer of 16 neurons. The input is the 9-dimensional MDP state and the output is the 4-dimensional action. We use tanh activation function and appropriately scale the network output to match the bounds of the action space. In total we have 208 weights which form our search space.

For initialisation, we searched over 100,000 seeds to check which gives the best initial policy. Many policies reached 1^{st} checkpoint(150 sec) and then stopped because of terminal condition. Out of those, we chose the one with the highest reward. Then we tried to hill climb on this initial policy using episode reward as the objective function but failed to do so for multiple combinations of hyperparameters and even with other initialisations that reached 150 seconds. In most cases, we couldn't climb even a single step and never crossed the first checkpoint.

The issue at play here is that all policies that were reaching the 1^{st} checkpoint had a high delta heading and altitude which triggered the terminal condition and hence searching around those policies did not allow us to move further. To solve this, we designed a new objective function that, at the cost of low reward, allowed us to move to a region in the search space where policies reached the first checkpoint with low delta heading and altitude. Our idea was that if we can reach such a point and then search around that, then we could actually cross the 1^{st} checkpoint and eventually get policies that gave even higher rewards. Our objective function is as follows:

Algorithm 1 Our objective function for hill climbing

```
Require: Climb starting at policy that reached $1^{st}$ checkpoint
  if $RunTime < 150s$ then
      $Objective \leftarrow 0$
  else if $RunTime$ is $150s$ then
      $Objective \leftarrow \frac{1}{|DeltaHeading|+1} \times \frac{1}{|DeltaAltitude|+1}$
  else
      $Objective \leftarrow EpisodeReward$
  end if
```

After running hill climbing with the new objective function, as expected, the reward initially decreased but the objective increased which allowed us to move to the desired regions in the search space and after a few more steps, we got policies that crossed the 1^{st} checkpoint and also gave a high reward. Finally, we also annealed the radius of the hypersphere to fine tune our search and get better policies. This led to the results tabulated in 5 and 6.

Table 3: Reward Weights

Reward Term	Weight
Heading	0.3
Altitude	0.3
Roll	0.1
Speed	0.1
Acceleration	0.2

Table 4: PPO Models

Model	Run Time(in sec)	Reward
Default Reward	134.083	234.899
Modified Reward	104.417	89.331

Table 5: Initial Policy Search (Radius of sphere = 1, No. of Samples = 10,000)

Step	Run Time(in sec)	Reward	Objective
0	150.00	648.32	0.000857
1	150.00	10.20	0.001353
2	150.00	116.66	0.002911
3	300.00	237.11	237.11
4	618.83	415.79	415.79

Table 6: Annealed Policy Search (Radius of sphere = 0.1, No. of Samples = 10,000)

Step	Run Time(in sec)	Reward	Objective
0	618.83	415.79	415.79
1	468.67	1729.61	1729.61

5 Conclusion and Future Work

We have implemented various algorithms in this work in the hope of solving the autonomous flight challenge. We have seen how PID controllers are not quite capabale of achieving sustained flights with varying target headings. We tried the PPO algorithm as it showed some promise, but realised in the end that the search space is so large that tuning PPO is a near-impossible task. Policy search yet again shines in this task, providing stellar results and flight-times with minimal training times. While our PID controller could barely produce 150 seconds of safe (as defined by the gym environment) and sustained flight, the policy search model was easily able to control the flight for over 600 seconds, providing over a $4\times$ improvement.
In order to extend this work, we would like to perform more extensive hyperparameter search and tuning on policy search and hill climbing. We would like to explore more deeply the Physics and Mathematics behind aircraft flight, and use this knowledge to handcraft even better state spaces. Optimizing over these state spaces using policy search may yield surprising results.

Acknowledgments and Disclosure of Funding

We thank Prof. Shivaram Kalyanakrishnan for his immense patience and continuous support for us throughout this project, and for providing us with various ideas including guiding us to policy search. We also thank our friend, Ms. Ayushi Gupta from the Aerospace Department at the Indian Institute of Technology Bombay, for her guidance on modern controllers used to pilot aircrafts like the A320.

References

Charles Audet and Michael Kokkolaras. Blackbox and derivative-free optimization: theory, algorithms and applications. *Optimization and Engineering*, 17, 02 2016. doi: 10.1007/s11081-016-9307-4.

Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 799–806, Cambridge, MA, USA, 2003. MIT Press.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Gordon Rennie. *Autonomous Control of Simulated Fixed Wing Aircraft using Deep Reinforcement Learning*. Department of Computer Science Technical Report Series. September 2018.

Wikipedia contributors. Pid controller — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=1080202662, 2022. [Online; accessed 5-May-2022].

J. G. Ziegler and N. B. Nichols. Optimum Settings for Automatic Controllers. *Journal of Dynamic Systems, Measurement, and Control*, 115(2B):220–222, 06 1993. ISSN 0022-0434. doi: 10.1115/1.2899060. URL https://doi.org/10.1115/1.2899060.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]