Deep RL based Nonlinear Adaptive Flight Control: on the gap between simulation and reality

Honours Programme Delft

Author: Adrian Beňo Supervisor: Dr. Erik-Jan van Kampen

Faculty of Aerospace Engineering, TU Delft, Netherlands

Abstract

Novel corrective algorithm bridging the gap between simulation and reality is proposed, which online fine-tunes an offline pre-trained deep reinforcement learning agent. The novel control architecture is inspired by the incremental model based heuristic dynamic programming. This novel control architecture is applied in an illustrative control environment. It was found that the corrective algorithm is able to help reach the desired reference state in an environment governed by different system dynamics than the system dynamics of the environment used during the pre-training of the reinforcement learning agent.

Introduction

The need for robust flight controllers has become apparent as the systems which must be controlled have substantially grown in complexity. Moreover, adaptability is one of the crucial properties of future flight controllers too. The base of this research is reinforcement learning (RL), which can handle complex control laws. A novel corrective algorithm is proposed to tackle the problem of adaptability.

Reinforcement learning

Reinforcement learning as a sub-field of machine learning is the "computational approach to learning from interaction" [1]. The learning phase composes of collecting reward R_{t+1} after taking action $\vec{a}_t$, given state $\vec{s}_t$. The environment then propagates the agent to the next state $\vec{s}_{t+1}$, at which point these steps begin to repeat in loop. This is schematized in Figure 1.

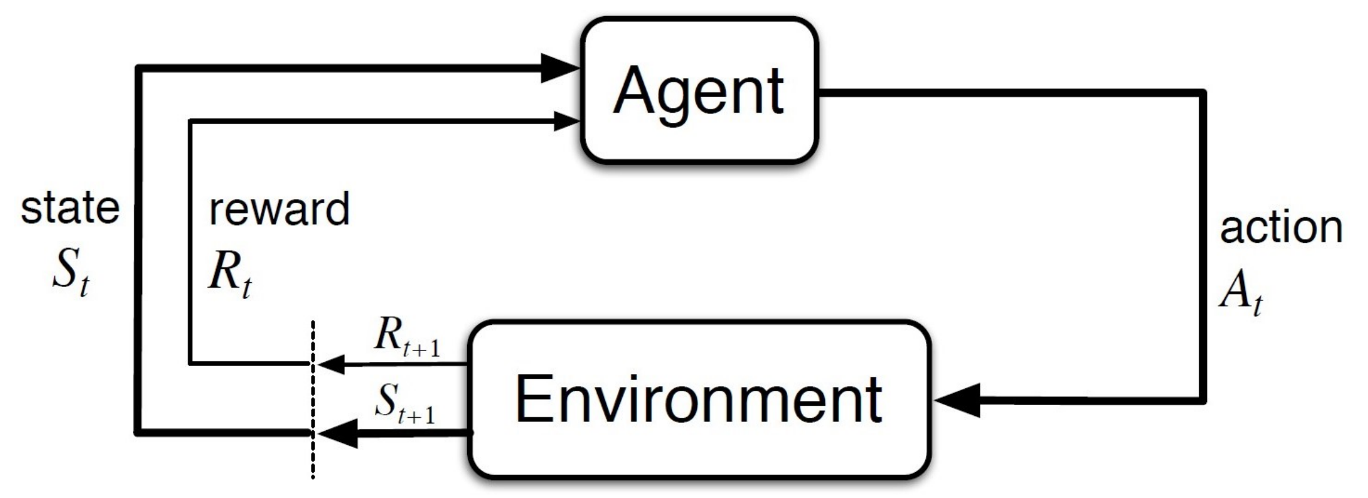

Figure 1. The learning scheme of an RL agent. [1]

The agent tries to learn a policy π, a mapping from state $\vec{s} \in S$ to action $\vec{a} \in A$, which maximizes the future cumulative collected reward $G_t = \sum_{t=t_0}^{T} R_t$. The learned policy can be approximated by any function approximator, such as a tabular method or an *artificial neural network* (ANN). In the latter case, the RL agent is said to be deep RL agent.

The corrective algorithm

The corrective algorithm consists of two networks: *learning actor* (L) and *target actor* (T). Initially, both are the same $\vec{w}_L = \vec{w}_T$ and correspond to the policy learned during the offline RL training. The learning actor then online updates its policy so that the expected outcome $\hat{x}_{t+1}$ of its action $\vec{u}_t$ in the real world approaches the outcome $\bar{x}_{t+1}$ of the target actor's action $\bar{u}_t$ in the training environment. Formally, the L2 norm e_a of the difference between $\hat{x}_{t+1}$ and $\bar{x}_{t+1}$ is minimized.

This is possible to do online due to an IHDP-inspired update rule [2], which informs the learning actor about the consequence of its actions via $\hat{x}_{t+1}$, supplied by the incremental model. The architecture of this corrective algorithm is given in Figure 2. The update rule of the learning actor is given in Equation 1.

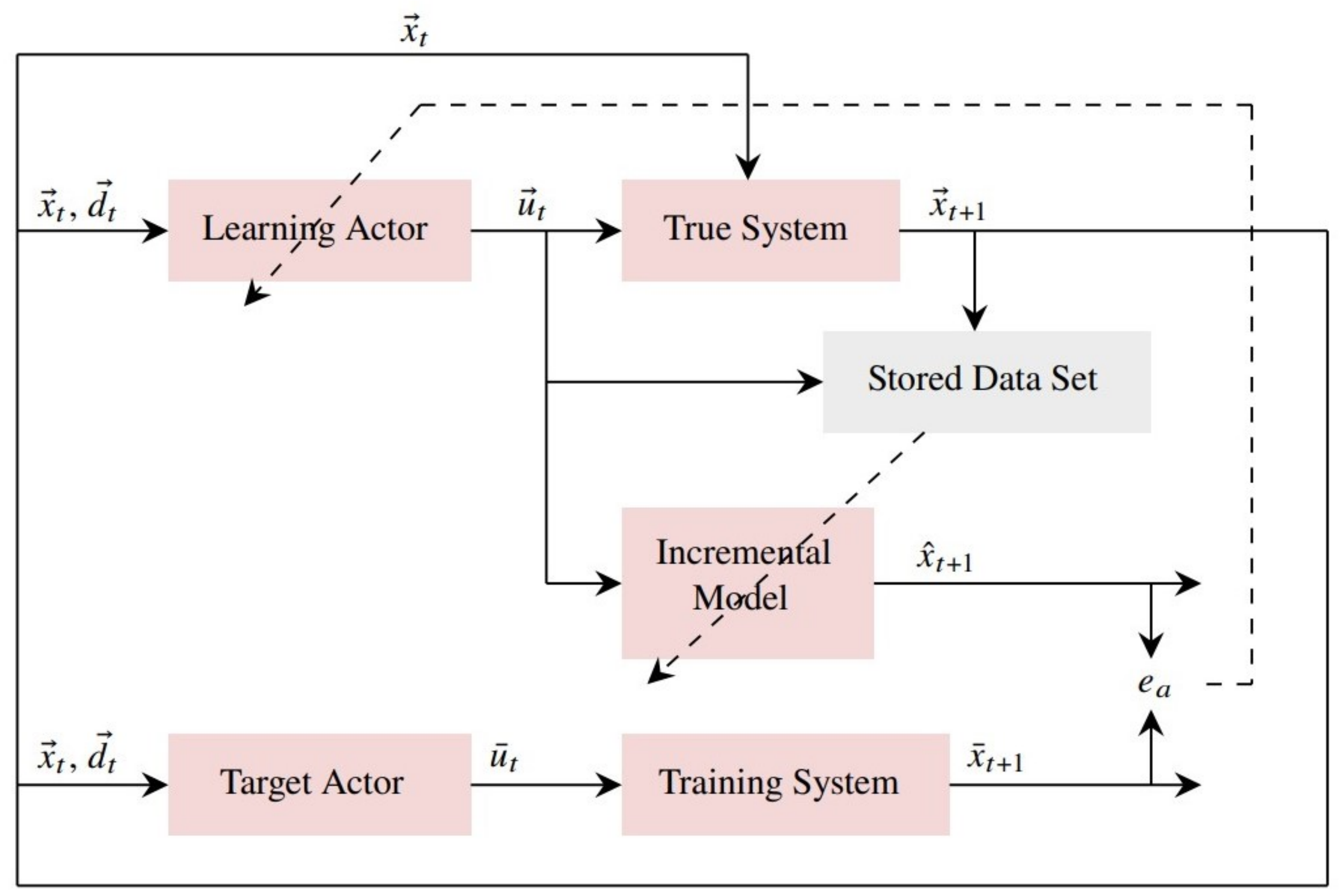

Figure 2. Corrective algorithm architecture. The training environment provides the states which shall be visited. The incremental model predicts the states which the agent will visit, given the current state and action. The learning actor then updates itself so that the state which it will visit is closer to the state which it shall to visit.

$$E_L = \frac{1}{2} e_a^2$$

$$\vec{w}_L \leftarrow \vec{w}_L - \alpha_L \frac{\partial E_L(t+1)}{\partial \vec{w}_L(t)} \qquad (1)$$

$$\frac{\partial E_L(t+1)}{\partial \vec{w}_L(t)} = \frac{\partial E_L(t+1)}{\partial e_a(t+1)} \frac{\partial e_a(t+1)}{\partial \hat{x}_{t+1}} \frac{\partial \hat{x}_{t+1}}{\partial \vec{u}_t} \frac{\partial \vec{u}_t}{\partial \vec{w}_L(t)}$$

Results

The performance of the corrective algorithm is evaluated in an illustrative environment - Gym's inverted pendulum environment [3]. Ultimately, we want to show that the corrective algorithm can online fine-tune the actions of the pre-trained RL agent, so that the trajectory it flies approaches the trajectory flown by the target actor in the training environment.

A small constant torque, $T_{rw} = -0.6$ [Nm], is added to the action chosen by the learning actor to simulate aerodynamical phenomena, which were not part of the training environment. The different trajectories are shown in Figure 3. It is clear that the corrective algorithm was able to almost exactly copy the target-actor-in-training-dynamics trajectory.

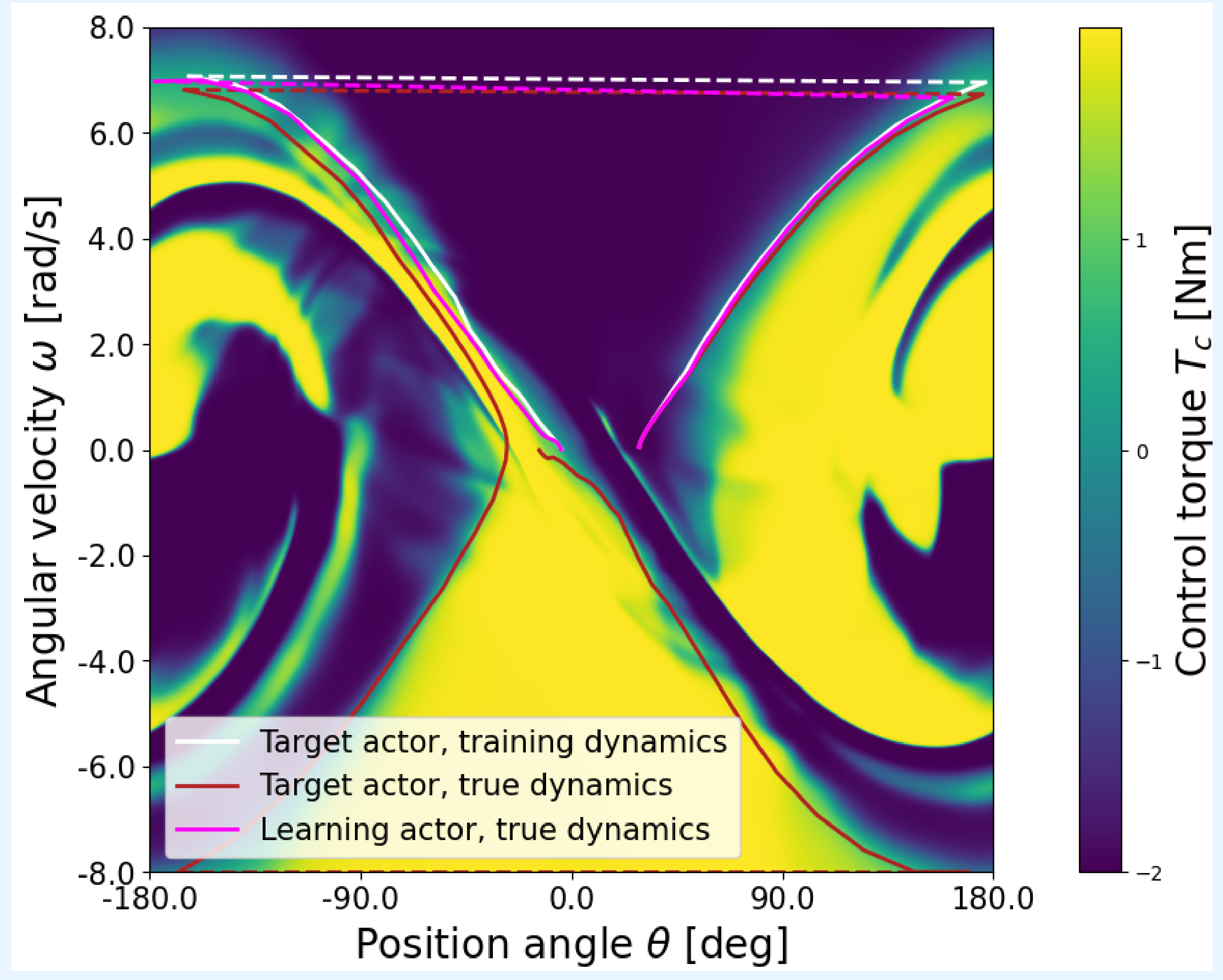

Figure 3. The trajectories with the actor's policy as the background. The dotted lines indicate the succession of two consecutive states. The dotted lines do not indicate the taken trajectories.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018, ISBN: 0262039249.

[2] Y. Zhou, E.-J. Van Kampen, and Q. Chu, "Incremental model based heuristic dynamic programming for nonlinear adaptive flight control,", Oct. 2016.

[3] G. Brockman *et al.*, *Openai gym*, 2016. eprint: `arXiv:1606.01540`.