ARL-TR-9603 • Oct 2022

DEVCOM ARMY RESEARCH LABORATORY

Learning Control Actions for Guided Projectiles Using the Proximal Policy Optimization (PPO) Algorithm

by Bethany Allik, Franklin Shedleski, and Christopher Hsu

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9603 ● OCT 2022

Learning Control Actions for Guided Projectiles Using the Proximal Policy Optimization (PPO) Algorithm

by Bethany Allik, Franklin Shedleski, and Christopher Hsu
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
October 2022	Technical Report	October 2021–August 2022

4. TITLE AND SUBTITLE
Learning Control Actions for Guided Projectiles Using the Proximal Policy Optimization (PPO) Algorithm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Bethany Allik, Franklin Shedleski, and Christopher Hsu

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLW-TC
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9603

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
primary author's email: <bethany.l.allik.civ@army.mil>.

14. ABSTRACT
In this report we present a data-driven approach for closed loop control of the Laboratory Technology Vehicle. We use the Proximal Policy Optimization (PPO) algorithm, a reinforcement learning algorithm that has been shown to perform well for a variety of tasks. The success of PPO is due to its stability in finding solutions, in addition to having many of the positive properties of policy gradient methods. Although PPO has been shown to be successful across the literature, it does suffer in the event of a sparse reward; this happens to be the case for our application of precision munitions where the objective is to hit a specific target. To tackle this issue of sparse reward we present a curriculum learning method around PPO. The curriculum segments learning into stages that incrementally increase in complexity, alleviating the sparsity of the reward signal. The proposed method is shown to outperform a learning method with no curriculum.

15. SUBJECT TERMS
Proximal Policy Optimization, PPO, control actions

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Bethany Allik
Unclassified	Unclassified	Unclassified	UU	27	19b. TELEPHONE NUMBER (Include area code) 410-306-0617

Contents

List of Figures

List of Tables

Acknowledgments

We would like to acknowledge James Maley for model development and Dr Mark Ilg for Functional Markup Unit compilation. Their help facilitated integration of the projectile model into the OpenAI Gym.

1. Introduction

Recently, data-driven methods for guidance, navigation, and control of autonomous systems have gained popularity. This is due to recent advances in machine learning, specifically deep learning and artificial neural networks. Reinforcement learning (RL) is a type of deep learning that aims to use interactions with the environment to learn an appropriate mapping from environment states to agent actions that maximizes the desired output. This procedure is inspired from natural processes, as most biological systems learn to operate in their environment through numerous actions with subsequent feedback. In RL, feedback from the environment is referred to as a reward signal. The system attempts to adjust inputs to maximize this desired reward signal. The inputs to the system are defined as agent actions, while the states and rewards are observed from the environment. These collected values are used to drive the learning process. In this work, we present an RL approach to developing a closed loop control scheme for a long-range precision weapon. The data-driven approach we use in this report is based on the Proximal Policy Optimization (PPO) RL algorithm.[1]

The rapidly evolving machine learning industry has led to new advances in RL that allow novel, data-driven approaches to control development. Even with highly dense inputs such as image frames,[2] actions can be inferred to maximize performance. Often times this approach makes closed loop control more straightforward, as in vision-based systems, where an image-based algorithm would not have to be developed separately and independently from the control. This unconventional approach goes against the traditional controller design, which is a model-based approach that relies on approximations of the system model. Approximations made due to parameter uncertainties and/or system nonlinearities often hinder the model-based approach, leading to underperforming or conservative controllers. For example, autonomous aerobatic flight is a challenging control problem because it requires precise control while operating on the edges of the flight envelope.[3] Although traditional, model-based methods can underperform when faced with extraneous situations, they do provide valuable performance guarantees against the known operational domain,[4] making them generally safe and predictable. Alternatively, model-free approaches require less model development and tuning to derive the closed loop control. Purely data-driven, model-free approaches can learn the intricacies of the system and can even scale in the number of agents used. However,

they require magnitudes more data[5] and performance guarantees in their control design can be more difficult to achieve.

RL approaches benefit from simplifications in the environment such as reward shaping or discretization of the action space and states to enable faster learning. In classical RL tasks, actions and rewards can be collected concurrently to continuously adjust the policy and maximize reward. Real-world problems are rarely posed in a way that allows this. For example, when training an autonomous agent to find the end of a maze, at every timestep there is no indication that the agent is applying the correct actions to the system until it has reached the time horizon or the goal. These types of problems are ill-fated to be defined with a sparse reward signal. To aid learning with a sparse reward, the designer could shape the reward to provide feedback continuously. This shaped reward has the disadvantage of inadvertently dictating the solution to the closed loop control, lessening the chances of finding an emergent solution obtained by allowing the agent to explore undirected. However, there is still merit to this approach when there is extensive domain knowledge that can be utilized.[6] Curiosity driven methods take the opposite approach by encouraging exploration into areas that are not well known. This has been demonstrated to work in numerous environments where curiosity is the only reward signal.[7] Another approach is to structure the system to learn progressively harder tasks to obtain the desired goal. This is referred to as curriculum learning, where the curriculum is a collection of progressively harder tasks that the system must learn incrementally.[8] The idea here is that the rewards will come more often in the beginning when the task is easy, providing the RL algorithm valuable feedback to be used to tune its controller.

2. Problem

RL has been implemented in numerous spaces including medical, social, and engineering applications. In this report, we apply RL to control a smart munition. Previous work on RL for missile guidance utilized reward-shaping methods to overcome the sparse reward problem. As already mentioned, this approach may cause the system to not explore paths that are not intuitive to the designer. Autonomous munition guidance, navigation, and control is a difficult task due to the highly uncertain and nonlinear dynamics of the projectile. Proportional navigation can be difficult to implement due to challenges in estimating line of sight rate [9] and time-to-go.

Proportional navigation is based on a linearized engagement geometry which may not be appropriate for the entire trajectory. Frequently this leads to ad-hoc decision points for switching from a "midcourse" guidance law and a "terminal" guidance law. Some of the difficulties in estimation stem from system nonlinearities, which force the control designer to make approximations and linearize the system. Some systems for projectile control use numerical differentiation of the flight equations,[10] which results in control decisions being made based on possibly erroneous states generated by noisy measurements. A data-driven approach may be advantageous for these systems.[11] However, the machine learning process is extremely difficult due to the sparse reward signal.

2.1 Contributions

In this report we present a method of applying RL to the difficult problem of closed loop control of a smart projectile. We designed an OpenAI gym environment that is embedded with a Functional Mockup Unit (FMU) model to closely simulate the real projectile. As such, this problem is more difficult than classical RL tasks due to the scale of the exploration task needed to find useful control policies. Here the states consist of position, velocity, and distance to target. The input action is the acceleration commands in the horizontal and vertical direction of the body frame. Due to the sparse reward in our problem a curriculum learning method is implemented, where stages of the curriculum align with larger to smaller "target" sizes. We show through experiments that with this system we can learn to fly a smart munition and hit a target with precision.

2.2 Organization

We introduce our environment simulation in Section 3, provide an overview of the PPO algorithm in Section 4, present our curriculum learning method in Section 5, give an overview of training in Section 6, then present our results in Section 7.

3. Architecture

The RL paradigm typically consists of a single or multiple agents interacting with an environment. At each discrete time step t each agent receives an observation of its state s_t from the environment. Agents will then select and inform the environment of an action a_t they wish to take. The environment finally calculates the outcome of agents' actions and grants each agent a reward R_t, completing the time step.

In our system we utilize the OpenAI Gym framework to orchestrate environment-agent interaction. A MATLAB model of the projectile generates the environment's state information (Cartesian position, body velocity, and line of sight information), which is provided to agents who then take y and z acceleration actions at each time step. OpenAI Gym handles learning and decouples it from the specifics of our environment model. This allows easy variance of hyper parameters and changing of learning algorithms. See Fig. 1 for a block diagram of our setup.

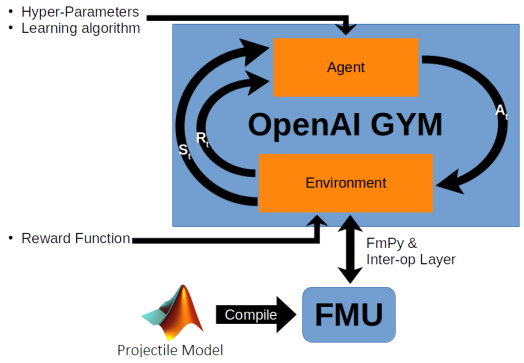

Fig. 1 OpenAI Gym provides the learning environment, fed by our FMU projectile model and curriulum reward signal, to multiple agents who learn using our selected algorithm, PPO, and hyperparameters

3.1 Model Integration

To bind the gym environment to the MATLAB model we are utilizing the functional mockup interface 2.0 (FMI2) standard. The FMI Toolbox in MATLAB was used to compile the model into an FMU. FMUs are cross-platform libraries that contain everything needed to solve their enclosed model. FMUs are also language agnostic, allowing the same model to be used in our Python learning as well as other research. This ensures our model is easily kept up to date as the overarching

program advances.

An inter-op layer utilizing FMPy, an open source project developed by Dassault Systèmes, is used to load the FMU and connect it to our gym environment. This allows leveraging both MATLAB's extensive modeling tools and the gym's RL framework. Utilizing FMPy and this inter-op layer streamlines updates to the model by decoupling the model's inner working from the environment. Updated FMUs can be incorporated into the learning system with zero code updates.

3.2 Model Monte Carlo Evaluation

The model is similar to that in Burchett et al.,[12] and is used in Fairfax et al.[13] The states in this model are the positions, velocities, and distance to target. The actions are horizontal and vertical acceleration commands in the body frame, which range $\pm 5.0g$. See Fig. 2 for sample trajectories given the extents of the control authority.

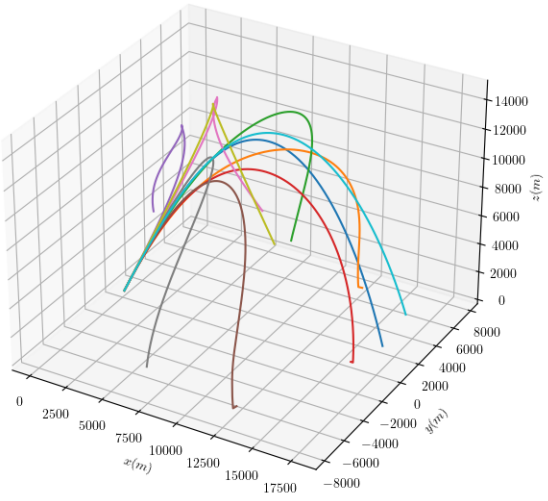

Fig. 2 Sample trajectories of the projectile. Maximum acceleration is applied for each direction to demonstrate the extents of the flight envelope.

4. Proximal Policy Optimization (PPO)

In this section, we introduce the PPO algorithm.[1] Policy optimization is an on-policy learning method set up such that the learning algorithm evaluates and improves the same policy by selecting actions and analyzing rewards. In other words, as the agent interacts with the environment simulator, we want to optimize the currently executed policy such that it receives a maximal reward at the end of a trajectory. In comparison to other policy optimization schemes, PPO strikes a balance between ease of implementation (in comparison to other policy gradient methods such as TRPO[14]), sample complexity, and ease of tuning. At each step, the algorithm computes a gradient and updates a policy such that it minimizes the cost function (or maximizes a reward) while ensuring the deviation from the previous policy is kept relatively small.

4.1 Policy Optimization

The primary idea of policy optimization is that we directly optimize for the thing we want, resulting in a more stable and reliable end product. In the case of policy optimization, we represent the policy explicitly as $\pi_\theta(a|s)$ where θ represents the parameters of deep neural network—they are optimized directly by gradient ascent on the performance objective $J(\pi_\theta)$ where this objective can be represented as a reward function.

Following the idea of optimization, we aim to maximize the expected return $J(\pi_\theta) = \mathbf{E}_{\tau\sim\pi_\theta}[R(\tau)]$ where $R(\tau)$ is the return or reward of the trajectory τ over some finite-horizon. With this objective, we can optimize the parameters of the policy via gradient ascent, for example, $\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\pi_\theta)|_{\theta_k}$, where α is the learning rate and the policy gradient is denoted as $\nabla_\theta J(\pi_\theta)$.

4.2 From Theory to Implementation

PPO theory takes significant effort to implement due to the current form of the policy gradient not being numerically computable. For example, reading the original paper for PPO and trying to implement it directly would not result in much success as there are many small tricks that are not mentioned in the theory but are essential in practice. Open source repositories such as Stable-baselines3[15] and Spinning Up[16] do a wonderful job of implementing clean code that is also well documented. In the case of this work, we utilize the code from Stable-baselines3, which borrows

the PPO implementation from Spinning Up. The following section is a summary of the Spinning Up implementation of policy optimization and the analytical policy gradient.

4.3 Spinning Up PPO Implementation

Continuing the story of needing an analytical version of the policy gradient, Spinning Up follows five tricks, to derive a new form for the expression.

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbf{E}_{\tau\sim\pi_\theta}[R(\tau)]. \tag{1}$$

1. Probability of a trajectory: The expanded form of the probability of the trajectory $\tau = (s_0, a_0, \ldots, s_{T+1})$ given that the actions come from π_θ is

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1|s_t,a_t})\pi_\theta(a_t|s_t).$$

With this in mind we want to expand the expectation in Eq. 1 and bring the gradient under the integral.

$$= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau)$$

$$= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) \tag{2}$$

2. The log-derivative trick: Based on the chain rule from calculus and given the derivative of $\log(x)$ with respect to x is $1/x$, we write:

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta).$$

This result allows us to expand Eq. 2 and return it to the expectation form.

$$= \int_\tau P(\tau|\theta)\nabla_\theta \log(P(\tau|\theta))R(\tau)$$

$$= \mathbf{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] \tag{3}$$

3. The last three tricks are ways in which to work out the gradient of the log probability of a trajectory; that is, $\nabla_\theta \log P(\tau|\theta)$. First, the log-probability of a trajectory, that is, writing out

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^{T} \big(\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t) \big).$$

4. The gradients of the environment functions that define the trajectory are zero as they have no dependence on θ so gradients of $\rho_0(s_0)$, $P(s_{t+1}|s_t, a_t)$, and $R(\tau)$ are zero.

5. Finally, the gradient of the log-probability of a trajectory is thus

$$\nabla_\theta \log P(\tau|\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t).$$

Putting it all together and continuing from Eq. 3,

$$\nabla_\theta J(\pi_\theta) = \mathbf{E}_{\tau\sim\pi_\theta} \big[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \big].$$

Since this is an expectation, we can estimate it with a sample mean. If we collect a set of trajectories $\mathcal{D} = \{\tau_i\}_{i=1,\dots,N}$ where each trajectory is obtained by letting the agent act in the environment using the policy π_θ, the policy gradient can be estimated with

$$\nabla_\theta J(\pi_\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)$$

where $|\mathcal{D}|$ is the number of trajectories in $\mathcal{D}$; that is, N. We showcase these steps to emphasize the difficulty in translating theory to practical implementation. Even after all this work, if $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is not computable, then this has all been for naught. Luckily, packages such as PyTorch and TensorFlow allow for the simple computation of gradients of a neural network using backpropagation and AutoGrad.

4.4 Key Points about PPO

The difficulty in implementing theoretical algorithms often makes it safer to use qualified implementations of state-of-the-art algorithms rather than to re-implement

them. However, using these algorithms can lead to pitfalls created by the small intricacies in their implementation required to allow them to work in practice. This section will highlight the main qualities of PPO, which make it our algorithm of choice.

PPO is motivated by a simple question: How can we take the biggest policy improvement step using the data we currently have without overstepping such that performance collapses? PPO improves upon TRPO, which tackles the same question but has a much more complex second-order method that is even more difficult to implement. PPO is a first-order method that uses a few tricks to accomplish the same goal of keeping policies close. The version of PPO that is implemented in Stable-baselines3 is called PPO-Clip. It relies on clipping in the objective function to keep the new policy close to the old one. In the original paper for PPO, the objective is given as:

$$L(s, a, \theta_k, \theta) = \min\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip}\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\Big) A^{\pi_{\theta_k}}(s, a)\Big).$$

Spinning Up simplifies this equation to be:

$$L(s, a, \theta_k, \theta) = \min\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), g(\epsilon, A^{\pi_{\theta_k}}(s, a))\Big),$$

where:

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A \leq 0. \end{cases}$$

A is the advantage function, which is the discounted rewards minus a baseline. It describes how much better or worse an action from the current policy is compared to all other actions on average. ϵ is a (usually small) hyperparameter, which roughly controls how far away the new policy is allowed to go from the old.

Even in the objective function, there is a major difference in how one can simplify to implement cleaner code. Clipping helps greatly to ensure reasonable policy updates, but in practice it is still possible to get a new policy that is too far from an old policy. Approaches differ between implementation. This version uses a simple method based on the mean KL-divergence of the new policy. If the mean KL-divergence of the new policy from the old policy grows beyond a given threshold, gradient steps

stop early.

Another important note is that PPO trains a stochastic continuous policy in an on-policy way, thus randomness can come in many forms. This randomness can be created by both initial conditions and the training procedure. Over the course of training, the policy becomes less random as the update rule encourages it to exploit rewards that it has already found. This is a feature and a bug as the policy might get trapped in local optima.

Due to the above features of PPO, the utilization of curriculum training is essential to successfully learning a policy for flying the LTV. Starting from a large target, when an optimal policy is found, we move to a smaller target and so on until we get to the desired target size. The algorithm's design to maximize stability necessitates that curricula be structured in such a way that transitions from policy to policy can be done without destabilizing.

5. Reward Signal

The reward signal is constructed to provide feedback for learning. For many aircraft systems there are certain constraints, other than the desired result, that need to be maintained as part of the flight control. These constraints depend on the application; for example, in Tang and Lai[17] an RL approach for landing of a fixed wing aircraft is presented. The vehicle has to follow a defined glide path. Thus altitude rate, horizontal velocity, pitch angle, and touchdown point outside the glide path lessen the reward. In Bøhn et al.[18] the reward function is designed to follow specific setpoints for attitude control of a fixed wing vehicle. For the missile application in He et al.[11] the reward function is shaped using the zero miss distance metric. This metric is the shortest distance between the missile and target from the time instant onward where both the missile and target do not maneuver.

Unlike these previous approaches we do not want to control the trajectory of the projectile, just guarantee success of the mission. To do this we maintain a sparse reward, which prevents inadvertently guiding the projectile. This allows the possibility of exploring more of the flight envelope. In typical long range missile guidance there are stages of flight; here we ignore this and hope the RL algorithm is able to learn a proper policy given the model and reward signal.

Due to the scale of our state space and sparsity of our reward we aid the learning process with a curriculum. At each learning level in the curriculum l the radius of the target gets smaller. Our simple reward is:

$$r^l = \begin{cases} 1 & \sqrt{(x^T - x_N)^2 + (x^T - y_N)^2} \leq D^l \\ -1 & \sqrt{(x^T - x_N)^2 + (x^T - y_N)^2} > D^l \end{cases}$$

where the value of r^l is the reward at level l, with munition positions x_N and y_N when it hits the ground at time, $t = N$. The position of the target is (x^T, y^T) and the radius of the target is D^l.

In practice this equation works best with another term that would reward flights with shorter durations. This would ensure engagements are done quickly and frivolous actions are not taken that would lengthen the flight. The total reward is:

$$r = r^l + \frac{C}{N}$$

where C is a scaling term and N is the trajectory length.

6. Training

This section describes the method used for training. Parameters are tuned for our specific problem of hitting a small, stationary target. The only goal for this munition is to hit said target. If training were structured to only reward the system when a target was hit, then the RL algorithm would most likely never learn. The exploration phase of an RL algorithm must stumble upon some rewarding behavior to be able to learn to maximize cumulative reward. Because our system has an inherently sparse reward compared to the input actions, we design the training to utilize a curriculum learning method to encourage beneficial behavior. The method decreases D_l at each stage of the curriculum, starting from an overly large $D_{l=L}$ to a desired $D_{l=0}$. The number of levels (L) are a training parameter and $D_{l=0}$ is a function of the target size and warhead size.

The following is the algorithm for our training method. In essence, we are running PPO with each level in the curriculum. Each PPO iteration is initialized with the previous policy weights.

Algorithm 1 Curriculum training method

input: env, levels
output: w
$initialize\ weights, w_0$
for $l \in \{1...L\}$ **do**
  $r_l \rightarrow env$
  $w_{l-1} \rightarrow w_l$
  Initialize PPO
  for $iteration = 1, 2, ..$ **do**
    PPO algorithm Reference 1
    if $r_l > \delta$ **then**
      $end\ training$
    end if
  end for
end for

7. Results

Here we present the results of our training method. To show the advantage of the curriculum learning method, we run the learning method with $L = 4$ levels and $L = 1$ levels. The levels indicate the radius around the target, with $l^{L=4} \in \{150, 100, 50, 5\}$ and $l^{L=1} \in \{5\}$. OpenAI Gym is used for this study, and the FMU is embedded as reported in Section 3. The PPO training parameters are in Table 1; information on the parameters is in Section 4 or Schulman et al.[1]

Table 1 Learning parameters for PPO

Hyperparameter	Value
time steps	2.4e5
step size	3e-3
Num. epochs	10
Discount (γ)	0.99
GAE Parameter (λ)	0.95

The $\delta = 1$ is set such that a significant number of evaluations of the current policy hit the target before moving to the next segment of the curriculum.

The LTV simulation is running at 5 Hz. The inputs are accelerations in the vertical and horizontal directions in the body frame. Note there is no stored power on the projectile; therefore, momentum in the axial direction (thust) cannot be ap-

plied. The acceleration commands are discretized to $\pm 5gs$ and the applied force is repeated $T = 10$ times. Repeating the input across multiple time steps not only speeds up training, but also restrains the system from performing unnecessary or jerky maneuvers.

At each stage of the curriculum method the learning is halted and 20 simulated flights are controlled with the learned policy, with 4 levels this provides 80 total runs for one training session. For the curriculum-less algorithm, 80 simulated flights are simulated with miss distance recorded. Miss distance is recorded for every simulated flight and is presented in Fig. 3, training sessions are repeated five times and the mean and standard deviation of the sessions is plotted versus trajectory simulation number.

As you can observe in Fig. 3, the curriculum-less case is never able to learn due to lacking an initial hit in the reward signal. The curriculum method is able to guide the solution and reduce the miss distance as the learning moves through the curriculum.

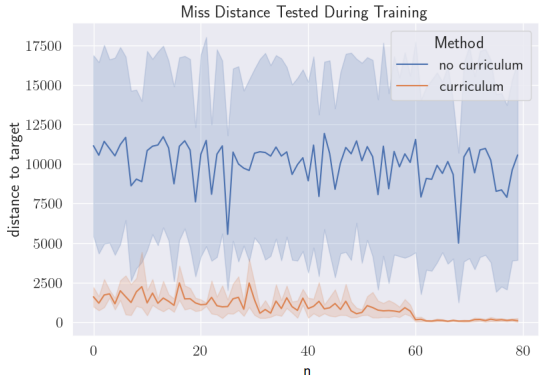

Fig. 3 Mean and standard deviation of miss distance at different levels of the curriculum compared to no curriculum method. There are $n = 20$ simulated flights recorded per stage of the curriculum and $n = 80$ simulated flights for the no curriculum method.

8. Conclusion

In this report we present an RL training method for the LTV projectile. The system has a sparse reward signal, which hinders training for all RL tasks. A curriculum training method is proposed, where stages of the curriculum represent incrementally smaller target radii.

We note that the single level case may improve via hyperparameter tuning. It is assumed that any improvement in parameter tuning would also benefit the multi-level approach, so the comparison is valid with identical hyperparameters in both cases.

9. References

1. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017.

2. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. 2013.

3. Clarke SG, Hwang I. Deep reinforcement learning control for aerobatic maneuvering of agile fixed-wing aircraft. In: AIAA Scitech 2020 forum; p. 0136.

4. Brunke L, Greeff M, Hall AW, Yuan Z, Zhou S, Panerati J, Schoellig AP. Safe learning in robotics: From learning-based control to safe reinforcement learning. arXiv preprint arXiv:2108.06266; 2021.

5. Hsu CD, Jeong H, Pappas GJ, Chaudhari P. Scalable reinforcement learning policies for multi-agent control. arXiv 2011.08055v4; 2020.

6. Grześ M. Reward shaping in episodic reinforcement learning. Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems; p. 565–573.

7. Burda Y, Edwards H, Pathak D, Storkey A, Darrell T, Efros AA. Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355; 2018.

8. Narvekar S, Peng B, Leonetti M, Sinapov J, Taylor ME, Stone P. Curriculum learning for reinforcement learning domains: A framework and survey. arXiv preprint arXiv:2003.04960; 2020.

9. Maley JM. Line of sight rate estimation for guided projectiles with strapdown seekers. Proceedings of the AIAA Guidance, Navigation, and Control Conference; p. 0344.

10. Fresconi F, Cooper G, Celmins I, DeSpirito J, Costello M. Flight mechanics of a novel guided spin-stabilized projectile concept. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering. 2012;226(3):327–340.

11. He S, Shin HS, Tsourdos A. Computational missile guidance: a deep reinforcement learning approach. Journal of Aerospace Information Systems. 2021;18(8):571–582.

12. Burchett BT, Paul JL, Vasile JD, Bryson J. A high fidelity roll dependent aerodynamic model for a long range, high speed missile. In: AIAA Scitech 2022 forum; p. 0420.

13. Fairfax L, Maley J, Bryson J. Collaborative delivery pacing requirement studies, part 1. DEVCOM Army Research Laboratory; forthcoming.

14. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P. Trust region policy optimization; 2015.

15. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research. 2021;22(268):1-8.

16. Achiam J. Spinning Up in deep reinforcement learning. 2018 [accessed 2022 Aug 30]. OpenAI.com.

17. Tang C, Lai YC. Deep reinforcement learning automatic landing control of fixed-wing aircraft using deep deterministic policy gradient. Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS); p. 1–9.

18. Bøhn E, Coates EM, Moe S, Johansen TA. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS); p. 523–533.

List of Symbols, Abbreviations, and Acronyms

FMI	functional mockup interface
FMU	Functional Mockup Unit
PPO	Proximal Policy Optimization
RL	reinforcement learning

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA

1 (PDF) DEVCOM ARL
FCDD RLD DCI
TECH LIB

1 (PDF) DEVCOM ARL
FCDD RLS EA
T GARNER
D WIKNER

1 (PDF) UNIV FLORIDA
W DAVIS

1 (PDF) DEVCOM C5ISR
J DEROBA
E BROWN

1 (PDF) DEVCOM AVMC
M HEIMBECK

6 (PDF) LOS ALAMOS NATL LAB
C BRONKHORST
R GRAY
A MISRA
D PRESTON
R LEBENSOHN
A SAXENA

5 (PDF) DREXEL UNIVERSITY
DEPT MEM
B FAROUK
N CERNANSKY
DEPT MATHEMATICS
P GRINFELD
DEPT MAT SCI & ENGNG
Y GOGOTSI
G TUCKER

2 (PDF) CALIFORNIA INST TECHNOLOGY
WA GODDARD, III
S ZYBIN

1 (PDF) CARNEGIE MELLON UNIVERSITY
A ROLLETT

9 (PDF) JOHNS HOPKINS UNIVERSITY
K HEMKER
N DAPHALAPURKAR
K RAMESH
J EL-AWADY
M FALK
L GRAHAM-BRADY
T HUFNAGEL
V NGUYEN
M ROBBINS

1 (PDF) UNIVERSITY MARYLAND
R ARMSTRONG

4 (PDF) MASSACUSETTS INST TECHNOLOGY
M DEMKOWICZ
R ABEYARATNE
L ANAND
D PARKS

3 (PDF) RUTGERS
G WENG
V DOMNICH
R HABER

1 (PDF) GEORGIA INST TECHNOLOGY
D MCDOWELL

1 (PDF) UNIV CALIFORNIA SAN DIEGO
MA MEYERS

1 (PDF) UNIV CALIFORNIA SANTA BARBARA
R MCMEEKING

ABERDEEN PROVING GROUND

105 (PDF) DEVCOM ARL
FCDD RLD P
T ROSENBERGER
FCDD DAS L
P BAKER
FCDD DAS LB
R BOWEN
FCDD DAS LBA
D FORDYCE
FCDD DAS LBD
R GROTE
L MOSS
J POLESNE
FCDD DAS LBE
M MAHAFFEY
C BARKER
R CIAPPI
C COWARD
D HOWLE
FCDD DAS LBG
P MERGLER
J ABELL
P HORTON
FCDD DAS LBS
M PERRY
D LYNCH
J SHINDELL
FCDD DAS LBW
R BOWERS
W MERMAGEN
FCDD RLW
S SCHOENFELD
FCDD RLW LA
W OBERLE
B BREECH
FCDD RLW LC
K MCNESBY
B ROOS
FCDD RLW LE
P WEINACHT
FCDD RLW LF
J CONDON
FCDD RLW LH
T EHLERS
E KENNEDY
L MAGNESS
C MEYER
B SORENSEN
R SUMMERS
FCDD RLW MB
B LOVE
G GAZONAS
FCDD RLW MC
R JENSEN
FCDD RLW MD
S WALSH
FCDD RLW ME
P PATEL
FCDD RLW P
D LYON
J HOGGE
T VONG
FCDD RLW PA
S R BILYK
P BERNING
J CAZAMIAS
M COPPINGER
J FLENIKEN
W C UHLIG
C WOLFE
FCDD RLW PB
C HOPPEL
M J GRAHAM
S SATAPATHY
FCDD RLW PC
M FERMEN-COKER
R BECKER
T BJERKE
D CASEM
J CLAYTON
M GREENFIELD
B LEAVY
J LLOYD
S SEGLETES
L SHANAHAN
A SOKOLOW
A TONGE
W WALTERS
C WILLIAMS

FCDD RLW PD
J RUNYEON
A BARD
N BRUCHEY
R DONEY
S HALSEY
M KEELE
D KLEPONIS
H W MEYER
R MUDD
F MURPHY
D PETTY
C RANDOW
S SCHRAML
B SCOTT
K STOFFEL
G VUNNI
V WAGONER
M ZELLNER
FCDD RLW PE
P SWOBODA
P BARTKOWSKI
D GALLARDY
D HACKBARTH
D HORNBAKER
E HORWATH
J HOUSKAMP
E KLIER
C KRAUTHAUSER
M LOVE
K MCNAB
D SHOWALTER
FCDD RLW PF
N GNIAZDOWSKI
C CUMMINS
E FIORAVANTE
D FOX
R GUPTA
S HUG
FCDD RLW PG
R FRANCART
S KUKUCK
C PECORA
J STEWART