# ME491 Learning-Based Control
# Final Project Report

## Anymal Sumo

*By Tye Cameron-Robson*

## Abstract

This report outlines the development and evaluation of a reinforcement learning-based agent trained to compete in a simulated robot sumo wrestling environment. The agent's objective is to knock over opposing robots or push them out of the circle in a constrained arena, as per the ME491 project specifications.

# Contents

# 1. Introduction

The task involves training a robotic agent in a simulated environment where the primary goal is to outmanoeuvre and defeat opponents with varying policies in a sumo-esque wrestling match. This project sought to combine elements of mechanical strategy with dynamic control, primarily by developing a reward function tailored to mechanical stability.

## 1.1. Problem Definition

The problem is formulated in an MDP (Markov decision process) framework, where the agent interacts with the environment in discrete time steps. Each step involves observing the state, taking an action, and receiving a reward based on designed performance metrics.

The state includes the robot's position, orientation, and velocity. Actions are the robot's limb movements, directly influencing its position and stability. The rewards are quantified based on stability, proximity to the opponent, and effective attacks, aligning with the objective of sumo wrestling.

## 1.2. Project Rule

- The control_dt and simulation_dt values must be 0.01 and 0.0025 respectively, for all opponents.
- The policy network must use MLP structure.
- Each opponent will play against one another 10 times:
    - 1 point is awarded for flipping the opponent
    - 1 point is awarded for removing the opponent from the arena
    - In the case of a draw, the robot closer to the centre of the arena at the 10 second mark will be awarded 0.01 points.

# 2. Methodology

## 2.1. Simulation Environment

Utilising RaiSim, the high-performance physics engine, a realistic simulation was initialised to mirror the dynamics of actual robotic sumo wrestling. The environment and constraints are strictly defined as per the project rules.

## 2.2. Agent Architecture

**Policy Network Structure**: An MLP (Multi-Layer Perceptron) structure was used with the architecture given in the “cfg.YAML” file. The policy network has two fully connected layers, each comprising 128 neurons. This network architecture was chosen for its effectiveness in mapping high-dimensional inputs to action spaces.

## 2.3. Learning Algorithm

The Proximal Policy Optimization (PPO) algorithm was selected for this project. It’s a popular approach in the field of reinforcement learning and is often implemented in robotic systems. PPO, developed by OpenAI, is an on-policy algorithm which optimizes an objective function to improve the policy (Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. 2017. Proximal Policy Optimization Algorithms. [online] Available at: https://arxiv.org/abs/1707.06347 [12/14/2023]). The main aspect of PPO algorithms is taking the biggest possible step to improve the policy without causing destructive large updates. This is practical for competitive scenarios using complex environments where large, abrupt policy updates can lead to critical performance malfunctions.

PPO achieves the balance by using clipping- or a clipped surrogate objective function. This function limits the policy update step, ensuring that the new policy adjustment is not too extreme when compared to the old policy. Specifically, it does this by clipping the probability ratio of the new and old policies, hence the name "Proximal". The clipping references a hyperparameter that controls how much the new policy can diverge from the old.

The primary advantage of PPO over other algorithms is simplicity and ease of implementation, while still maintaining robustness and efficiency. PPO removes the need for a second-order optimization process, reducing computational costs. Additionally, PPO is more sample efficient compared to algorithms like Deep Q-Networks (DQN), making it more suitable for complex environments where obtaining large amounts of data is costly or impractical.

When regarding the project's objective, PPO is particularly well suited- train the agent to knock over opposing robots or push them out of the circle. The task involves a high-dimensional state space and requires a delicate balance between exploration and exploitation, PPO is known for handeling this well. Its ability to constrain policy updates ensures stable learning, avoiding drastic behavioural changes that could be detrimental in a competitive setting. The nature of the task, where small changes in strategy can lead to a critical failure or a loss in the competition, makes PPO's cautious optimisation approach advantageous.

Furthermore, the PPO algorithm works with continuous action spaces allowing for smooth and precise control of the robot's movements, a large factor in outmanoeuvring opponents in the arena. This precision, combined with the stability of the algorithm, reinforces PPO as a valid choice for this project.

A unique yet simplified approach to curriculum learning was employed to enhance the training of the reinforcement learning agent. Curriculum learning typically involves gradually introducing more complex tasks or environments to the learning agent. However, in this implementation, the simple curriculum is embedded within the reward structure, a method particularly well-suited for the environment.

There were 2 stages to the curriculum:

- When the distance is larger than 0.5m from the centre:
    - When the agent is further from the centre, there is an emphasis on accelerating towards the opponent. The impact reward function is greater and the change in distance reward is measured from the opponent, as opposed to the centre. The forward velocity function is also multiplied by 2.5x
    - Stability and height maintenance rewards ("stable1" and "stable2") are maintained at standard levels.
- When the distance is less than 0.5m from the center:
    - As the agent approaches the center, the reward structure shifts to emphasize precision and control.
    - The reward for maintaining a position close to the center ("staticDist") is scaled with the timer, emphasising the importance of this objective as the match progresses.
    - Stability and CoM height rewards are increased (multiplied by 2.5), underscoring the need for balance and control when in direct conflict with the opponent.

# 3. Experiments and Results

## 3.1. Training Process

The project utilized a custom environment defined in the ENVIRONMENT class, which involved simulating an "anymal" robot and box with similar yet simplified dimensions as the opponent in a competitive arena. The actor-critic architecture was used, with a network for the policy and a network for the value. Both networks' structures were defined in the YAML configuration file as containing two hidden layers of 128 neurons for each network. The Actor (policy network) is responsible for choosing actions in the environment and the critic (value network) is used to estimate the state values, and hence long-term rewards of those states.

Training the movement and strategy started very unstable, but as more rewards were implemented and the parameters tuned there was a visible increase in the refinement of the movement and strategy developed around the unique environment.

The main challenge was with processing power. The more rewards that were accounted for during training, the slower the simulation would run. Similarly, when implementing an opponent with a policy network there was a large impact to simulation speed. The idea to overcome this was a very simplified model box which replicated the movement of the anymal.

This series of variable parameters relating to the simulation and reward functions were adjusted constantly through the training phase:

- **Environment number** shows the number of parallel environments used for training, used to control the speed of data collection and diversity of actions per step. It was set as 100 to balance speed and computation cost. While adjusting other hyperparameters, it was reduced purely for speed of computation.
- **Evaluation Frequency** is used to monitor the performance of the training. Similarly, it was reduced when tuning the parameters.
- **Thread number** always set as 30 to achieve maximum CPU utilisation
- **Simulation Time Step** determined the granularity of physics calculations. This remained at 0.00025 seconds for the same level of simulation accuracy for all opponents.
- **Control Time Step** is the interval which the policy is updated. It was kept at 0.01 seconds, providing 100 actions per second.
- **Max Time** of the simulation was set to 10 seconds. This was to follow suit with the project rule of termination at 10 seconds if no terminal state was met.
- **Action Std.** is the standard deviation used in the action space, tuned for exploration purposes. It remained at 0.1.

| Environment Number | Evaluation Frequency | Thread Number | Simulation Timestep | Control Timestep | Max Time | Action Std. |
|---|---|---|---|---|---|---|
| **100** | **200** | **30** | **0.00025** | **0.01** | **10.0s** | **0.1** |

Other parameters such as the rewards function coefficients of the simulation were also tuned:

- **Forward velocity** coefficient controlled the reward for the anymal for moving in the forward direction which distinguished the front of the robot from the back and sides.
- **Torque** coefficient negatively rewards the anymal for using too much joint torque- improving the stability of motion.

- **Useful distance** coefficient is used to control the reward generated whenever the anymal was moving towards the current target (i.e. opponent, centre of the arena).
- **Impact reward** coefficient controlled the reward determined by contact with the opponent.
- **Static Distance** coefficient controlled the inversely proportional reward determined by the distance from the centre of the arena.
- **Stabliser 1** coefficient controlled the reward given by the Height of the CoM of the anymal body- a lower centre of mass to provide more stability.
- **Stabliser 2** coefficient controlled the reward determined by the vertical rotation offset of the anymal's body, to prevent flipping during battle.

| Forward Velocity | Torque | Useful Distance | Impact reward | Static Distance | Stabliser 1 | Stablser 2 |
|---|---|---|---|---|---|---|
| 0.2 | -4e-5 | 1000 | 0.2 | 0.001 | 0.2 | 0.2 |

The final parameters set were PPO-specific parameters:

- **Discount factor** (gamma): 0.998
    - This parameter determines the importance of future rewards. This remained set very high to encourage the simulation to achieve a good final position at the end of the 10 seconds and to ensure that fatal mistakes were not made with the intention of a strong immediate reward.
- **GAE parameter (lambda)**: 0.95
    - This parameter remained at 0.95 for a similar reason as the large discount factor. A balanced long and short-term advantage prevents fatal mistakes for the sake of immediate or future rewards.

## 3.2. Performance Evaluation

### Quantitative Analysis:

The robot placed 24 out of 28 competitors, with ~15% winrate and minimal successful draw rate.

### Qualitative Analysis:

- The general behavioural observations of the robot were erratic joint movements but a stable main body.
- The most notable competitive behaviour of the robot was an initial leap at the opponent, with the assumed objective to have a large impact and initially jar the opponent's stability. The times where this was successful earned the 15% win rate, as the opposition robots would lose due to ground contact with the body very early on.
- In the case of draws, the initial leap would destabilise the opponent's robot sufficiently and they would lock into a force competition to attempt a finish closest to the centre. In training, the opponent box exerted the same movement as the robot, such that the robot would be pushing against itself to remain in the centre. This is likely the training which gave way to the successful draws.
- The losses were often caused by the initial leap backfiring, causing the robot itself to destabilise and the body to collide with the floor. This wasn't picked up in training due to the initial collision with the opponent having little variance, as it would always demonstrate the same behaviour as the robot itself.

- Due to the narrow environment and limited opponents during training, the policy would reach an optimal state at around the 500th iteration, limiting room for improvement.

### 3.3. Key Contributions

- Unstable joint torques
- Initial leap- positive when effective, but often backfired.
- Strong body stability
- Narrow training environment

## 4. Discussion

### 4.1. Insights and Learning

- Erratic Joint Movements:
    - Despite having a stable main body, the robot displayed erratic joint movements, which is likely due to the error when scaling the joint torque coefficient with the other coefficients.
- Initial Leap Strategy:
    - The robot's strategy of an initial leap towards the opponent was a double-edged sword. When successful, it destabilised the opponent leading to early wins. However, it also frequently backfired, causing self-destabilization.
    - Without this strategy, it is also likely for many more successful draws to occur.
- Draws in Tightly Contested Matches:
    - Draws were achieved in situations where both robots engaged in a forceful struggle near the centre. This indicates that the robot had learned to exert sustained force, a likely result of its training against a similarly behaving opponent (the box).

### 4.2. Limitations and Improvements

The ultimate limitation of the simulation was the opposition model. Modelling a box of similar dimensions and similar movement to the robot itself limited the scope of training. This highlights the importance of training in diverse environments. Introducing a range of opponent behaviours and scenarios in the training phase could better prepare the robot for the unpredictability of real-world competition.

The aggressive leaping approach likely caused more harm than benefit. When discovering these behaviours, there should be a measure and reward based on the risk of failure. This can be more easily implemented and measured in training environments with varying opponent behaviour.

The erratic movement of the joint angles shows that when varying reward coefficients, changes can affect the weight and effect of other coefficients. Hence the coefficients required much more focused tuning.

The impact reward was practical, but it encouraged the robot to maintain constant contact with the opponent, as opposed to discovering movements which would cause the opponent to lose. This reward function would need revising such that the anymal is rewarded for destabilising the opponent, as opposed to just making contact.

## 5. Conclusion

This project's progress in implementing a trained policy for a sumo-esque competition has given valuable insights into the practical implementation and challenges when applying learning-based control in dynamic environments. The primary takeaway is the pivotal role of strategic training and

fine-tuning in reinforcement learning, especially in a competitive and dynamic context like robot sumo wrestling.

The importance of a careful balance in approach between exploration and exploitation was demonstrated by the use of a Proximal Policy Optimisation (PPO) algorithm. The project highlighted strengths when using PPO, particularly its ability to handle state spaces with large dimensions while ensuring stable learning progressions. However, the sensitivity of such systems to hyperparameter settings and reward structures has also been noted.

The implementation of a very simple curriculum learning approach within the reward function was used to adapt to the environment's competitive nature. This approach focused on varying strategies based on the robot's proximity to the center, showcasing the potential of curriculum learning in competitive instances of reinforcement learning. However, the results also showed a requirement for a more developed implementation, in particular a consideration of the risk-reward balance of aggressive strategies like an initial leap tactic.

One of the critical learnings from this project is the importance of diversity in training scenarios. The use of a simplified opponent model, although beneficial for computational efficiency, ultimately limited the scope of the learning, leading to a lack of adaptability in the simulation. This shows the need for more complex and varied training environments in the future to prepare agents for unpredictability and varied strategies. In this scenario, generating a policy which can deal with opponents defending and attacking would be the next objective.

In conclusion, this project advanced my understanding of applying reinforcement learning to control robots in situations which normally would be very difficult to model in typical control engineering. It highlighted the involvement between the PPO algorithm selection, hyperparameter tuning, training environment variety, and the formulation of reward functions. Future work in this area can build upon these learnings, with focuses on diversified training environments, developed reward structures, and exploring any potential limitations of learning-based control in competitive settings.