PROXIMAL POLICY OPTIMIZATION FOR MATCHING TARGET ORBITS

1st Carlos Chavez
ASEN Masters Student
University of Colorado Boulder
carlos.chavez@colorado.edu

2nd Ashwin Raju
ASEN Masters Student
University of Colorado Boulder
ashwin.raju@colorado.edu

3rd Yusif Razzaq
ASEN Masters Student
University of Colorado Boulder
yusif.razzaq@colorado.edu

Submission to: Dr. Zachary Sunberg, University of Colorado Boulder, ASEN 5354 Decision Making Under Uncertainty, Ann and H.J. Smead Aerospace Engineering Sciences College of Engineering and Applied Sciences.

I. Introduction

The primary objective is to develop an reinforcement learning(RL)-based framework capable of generating transfer trajectories that navigate between two specified periodic orbits while satisfying mission-specific objectives. This framework will consider multiple objectives, such as minimizing fuel consumption and minimizing position deviation from desired orbits. Although motivated by Sullivan's, Bosanac's, Mashiku's, and Anderson's work on Multi-objective RL optimization under the circular restricted three body (CR3B) context [1], after analysis of CR3B and multi-body systems [5], the project is significantly simplified to only consider planar orbits under under the two body context.

II. Preliminaries

A. Two Body Dynamical Model

The two-body dynamics model makes several simplifying assumptions to reduce the complexity of the problem. These assumptions include considering only two significant bodies (the spacecraft and Earth) and assuming that Earth is a perfect sphere with uniform mass distribution. Under this model, the only force acting on the spacecraft is the gravitational force:

$$\mathbf{F}_g = -\frac{G \cdot M}{r^3} \cdot \mathbf{r}$$

where G and M are constants, and $\mathbf{r}$ is the position vector defined in the Earth-centered inertial (ECI) frame. The state of a spacecraft in orbit can be described completely by a six-element state vector that can take any real value:

$$\mathbf{s} = [r_x, r_y, r_z, v_x, v_y, v_z]^T \in \mathbb{R}^6$$

where r and v position and velocity components. A linear dynamics model can then be properly defined:

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{r_x} \\ \dot{r_y} \\ \dot{r_z} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ -\frac{G \cdot M}{r^3} \cdot r_x \\ -\frac{G \cdot M}{r^3} \cdot r_y \\ -\frac{G \cdot M}{r^3} \cdot r_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta v_x \\ \Delta v_y \\ \Delta v_z \end{bmatrix}$$

The control inputs to the system include an instantaneous ΔV burn. This dynamics model was used to simulate the orbit's propagation through time along with the influence of the applied controls.

B. Environment Representation

To facilitate the application of RL techniques to the problem, a proper representation of the environment was formulated. The state space maintained a continuous representation, corresponding to the six-element state vector previously defined. A two dimensional action space was considered, consisting of a element to describe the time duration to wait, as a fraction of the current orbital period, and the an element corresponding to the magnitude and direction of the thrust. These actions where discretized within a finite interval, and a set of discrete actions were produced from the Cartesian product of each dimension:

$$A_w = \{0, \Delta t, \Delta t, \ldots, 1 - \Delta t\}$$

$$A_T = \{T_{min}, T_{min} + \Delta T, \ldots, T_{max}\}$$

$$\mathcal{A} = A_w \times A_T$$

$$a = (a_w, a_t) \in \mathcal{A}$$

There was assumed to be no uncertainty in the state transitions. Therefore, the transition function can be defined according to the dynamics equations and selected actions:

$$T(s' \mid s, a) : s' = s + \int_0^{a_w} f(s, a_t)\, dt$$

Various different reward functions were tested throughout the course of this project, becoming a topic of considerable interest in attempting to improve convergence time and performance.

C. Deep Reinforcement Learning

Proximal Policy Optimization (PPO) is a RL strategy designed to train policies in complex environments where the action space can either be discrete or continuous. PPO's main features include:

1) **Actor-Critic** The structure trains two neural networks to uncover locally optimal behavior within the environment.

Actor Network: The actor component of the architecture is responsible for proposing actions given the current state of the environment. the actor maps optimal actions to every state in the environment, denoted as the policy function $\pi(s_t\ u_t)$:

- For **continuous spaces**, it outputs actions through a series of transformations capped by a tanh function, ensuring that the actions are bounded.

Critic Network: The critic component evaluates the potential value of the state provided by the environment, which guides the actor's decision-making. It estimates the value function, $V\pi(st)$, for every state in the environment. It consistently employs a series of transformations followed by a tanh function, culminating in a single output that estimates the state value.

2) **Clipped Surrogate Objective** The objective helps training stability by limiting the amount a policy can deviate from an old update to a new update. It avoids detrimental policy updates from overly optimistic estimates of the trust region surrogate objective by *clamping*.

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

where $r_t(\theta)$ is the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},$$

and $r(\theta_{\text{old}}) = 1$.
ϵ is a hyperparameter, say, $\epsilon = 0.1$. The motivation for this objective is as follows. The first term inside the min is L_{CPI}. The second term, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$. Finally, we take the minimum of the clipped and unclipped objective, so the final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective [2].

3) **Multiple Epochs of Mini-Batches** PPO uses data across multiple updates(epochs), allowing the policy to learn from existing and new data. It grabs mini-batches from the data to update the policy, which manages computational expenditure.
4) **Discrete or Continuous Action Spaces** It evaluates both discrete and continuous actions space, making PPO suitable for maneuvering a space craft.
5) **Advantage Estimation** The advantage function is enforced to evaluate how beneficial a state-action pair is based on the expected and returned rewards.

III. Problem Formulation

A. PPO Parameter Selection

Using the PyTorch package within Python [4], the PPO algorithm implementation underwent testing using different sets of hyperparameters. Table I reflects the resulting parameters and their respective values utilized for policy optimization in our final approach.

TABLE I
PPO Hyperparameter and Construction Parameters

Quantity	Value
Update Phases	40
Environmental Steps, τ	1600
Epochs, E	40
Mini Batches, M	4
Discount Factor, γ	0.99
Value Function Coefficient, c1	0.5
Entropy Coefficient, c2	1×10^{-2}
Clipping Parameter, ϵ	0.2
Actor Learning Rate, lr	3×10^{-4}
Critic Learning Rate, lr	1×10^{-3}
Actor Hidden Layers	2
Actor Nodes per Hidden Layer	64
Critic Hidden Layers	2
Critic Nodes per Hidden Layer	64
Activation Function	Tanh

B. States, Actions, and Reward Formulation

*1) **State Representation**:*

- **State Space** $\mathcal{S} = \mathbb{R}^6$
- **Initial Orbit** The spacecraft was initialized at a circular orbit about the Earth with an altitude of 300 km.

$$\mathbf{s}_0 = [6671 \text{ km}, 0, 0, 0, 7.7298 \text{ km/s}, 0]^T$$

- **Goal Orbit** The desired orbit was chosen to be a geosynchronous circular orbit at an altitude of 35786 km.

$$\mathbf{s}_f = [42157 \text{ km}, 0, 0, 0, 3.0749 \text{ km/s}, 0]^T$$

*2) **Actions**:* $\mathcal{A} = A_w \times A_T$

*3) **Reward Function Considerations**:*

- **Sparse Rewards** A sparse reward function refers to a situation where an agent receives a reward only occasionally or in specific circumstances. This may be useful in providing incentives for the agent to achieve a desirable configuration such as particular terminal state. However, sparse rewards can make the learning process more difficult because the agent has less information about its performance as it performs action in the environment.
- **Dense Rewards** A dense reward function provides the agent with frequent rewards throughout the learning process, giving the agent feedback at every time step or action, guiding it towards the desired behavior more directly. When implemented properly, dense reward functions can significantly speed up the learning process by providing the agent with more immediate and precise information about its performance. However, it is possible to encode undesirable behaviors using this representation, as the optimizing agent may find the means to exploit the intermediate reward and circumvent the need to fulfill high level specifications.
- **Reference Trajectory with Spare and Dense Rewards** The reward function is designed to guide a spacecraft from an arbitrary initial circular orbit to a specified final circular orbit using a Hohmann transfer trajectory. The reward is calculated based on the spacecraft's proximity

to the initial, final, or reference (Hohmann transfer) trajectory. It aims to encourage the spacecraft to follow the planned trajectory while minimizing propellant usage.

- **Initial Orbit** If the spacecraft is closest to the initial orbit, the reward incentivizes staying near the initial orbit parameters while considering propellant efficiency.
- **Reference (Hohmann Transfer) Trajectory** When the spacecraft is closest to the reference trajectory, the reward prioritizes following the Hohmann transfer path as closely as possible while also minimizing fuel consumption.
- **Final Orbit** If the spacecraft is within a set range of the final orbit (e.g., 5% difference in orbital elements), the reward prioritizes guiding it towards reaching the final orbit accurately while still reducing propellant usage

The reward function also incorporates penalties proportional to deviations from the desired distance and velocity changes across the three phases.

IV. Matching Final Orbit with Different Reward Functions

A. Environment Setup

The orbit transfer problem was formally addressed within the framework of OpenAI's Gym API [3]. This involved the initialization of a class with specified action and observation spaces. Given our assumption of perfect state knowledge, the observation space mirrored the state space. Additionally, we implemented a STEP() method to advance the dynamics model based on selected actions and return the resulting state, reward, and termination condition. A RESET() method was also defined to ensure each episode started from the initial orbit. With the environment established, we employed a PyTorch implementation of PPO [4], utilizing a clipped objective function to train within the Gym environment. This framework offered a straightforward yet reliable means to validate our environment definitions and make adjustments as needed.

B. Approach 1: Single Sparse Reward

To ensure the environmental setup and PPO algorithm were performing as expected, a simple formulation of the problem was considered. The action space was reduced to only consider a finite set of thrust magnitudes applied every half-period. Effectively, this reduces the wait action space to a singleton, $A_w = \{0.5\}$, which constrains the spacecrafts ability to induce a ΔV to align with the orbit's semi-major axis. Since the starting and final orbit were circular, this would ensure that the optimal Hohmann transfer could be achieved. The trust magnitude and direction were chosen jointly from the interval [-1, 3], discrete in 200 unique actions.

$$\mathcal{A} = \{0.5\} \times \{-1, -1 + \frac{3+1}{200}, \ldots, 3\}$$

A sparse reward function was used to award the agent only upon reaching the desired orbit, which would terminate the episode. At each step, the agent would incur a cost proportional to the magnitude of the ΔV burn applied. Additionally, if the agent collided with the Earth or achieved an escape velocity, a large negative reward would be assigned and the episode would terminate. Formally, the reward function for the simplified problem was defined as follows:

$$R(s,a) = R_{fuel}(a) + R_{\text{terminal}}(s)$$

$$R_{fuel}(a) = -50 \cdot |\Delta V|$$

$$R_{\text{terminal}}(s) = \begin{cases} 2000 & \text{if } \text{diff}_a < 0.05 \ \& \ \text{diff}_e < 0.05 \\ -2000 & \text{if } sa < r_{Earth} \end{cases}$$

where,

- $\text{diff}_e = |(e - e_f)/e_f < 0.05|$
- $\text{diff}_a = |(sa - sa_f)/sa_f < 0.05|$

where sa is the semimajor axis of the current orbit.

C. Approach 2: Sparse Reward with Expanded Action Space

The second approach simply modifies the action space back to the original problem formulation, allowing the agent to perform maneuvers at four locations in an orbit. This increased the size of the action space to 800, aiming to provide the agent a more expressive capacity for maneuvering between orbits.

$$A_w = \{0, 0.25, 0.5, 0.75\}$$

$$\mathcal{A} = A_w \times A_T$$

$$a = (a_w, a_t) \in \mathcal{A}$$

This approach maintains the sparse reward as described in Approach 1. However, the increased action space presented the need for reward shaping in order to improve convergence to the optimal policy.

D. Approach 3: Dense Linear Reward Function

In order to improve the convergence time when considering large actions spaces, adjustments to the reward functions were made to incentivize exploration of promising regions of the state space. This approach established a continuous reward function defined over the entire state space that would be reward the agent according to its compliance with the desired orbital elements. The reward components for the alignment of semi-major axis and eccentricity can be structured as an alignment between the current orbit's orbital elements and the desired orbit's orbital elements by the following linear reward function:

$$R_{\text{linear}} = 100 \times \begin{cases} \max(0, 1 - \text{diff}_a), & \text{(Semi-major Axis)} \\ \max(0, 1 - \text{diff}_e), & \text{(Eccentricity)} \end{cases}$$

The completion reward, $R_{terminal}$, and the penalty for fuel consumption, R_{fuel}, remain unchanged.

E. Approach 4: Hohmann Reference Reward Function

The reward function for the Hohmann Transfer Trajectory simulation is structured to penalize the deviation of the spacecraft's state from the reference trajectory in addition to the consumption of fuel.

$$R = \max\left(-100, -0.01 \times \ (\Delta_{r_h} + \Delta_{v_h} - |\Delta V|)\right)$$

where,

- Δ_{r_h} is the Euclidean distance between the current and expected position vectors from the Hohmann Transfer at specific time.
- Δ_{v_h} is the Euclidean distance between the current and expected velocity vectors from the Hohmann Transfer at specific time.
- ΔV is the magnitude of the burn applied.

The reward is bounded below by -100 to ensure that the penalty does not become excessively negative, promoting stability in the learning process. Once the Hohmann transfer trajectory in reached, the reference orbit is replaced by the goal orbit, and costs are assigned with respect to deviations from it. The sparse reward for achieving the goal orbit was maintained as in the previous approaches.

V. Results

A. Approach 1: Single Spare Reward Results

The reinforcing learning agent was able to successfully learning the policy for optimal orbit transfer between two co-planar orbits. As depicted in Figure 1, this policy was derived in approximately 60,000 steps in the environment using the reduced action space of 200 discrete thrust control inputs. This showed the potential efficacy of utilizing RL techniques to solve trajectory design problems.

The policy was then evaluated to reconstruct the spacecraft's trajectory for a representative episode, shown in Figure 2. The initial orbit is shown in green, the trajectory taken by the spacecraft in purple, the final orbit at termination in orange, and the desired orbit in red. This trajectory closely models a Hohmann transfer, which is know to be the most efficient means of performing a co-planar orbit maneuver.

B. Approach 2: Sparse Reward with Expanded Action Space

When expanding the action space in the same PPO implementation, the algorithm struggled to converge onto an optimal policy, leading to a noticeable failure to effectively learn as depicted Figure 3. A larger action space inherently increases the complexity of the learning problem. With more possible actions to choose from, the agent needs to explore a larger solution space, making it more challenging to find the optimal policy. Thus, finding meaningful patterns and generalizing across actions becomes more difficult, leading to suboptimal policy decisions.

Despite the poor learning performance, the single best trajectory witnessed throughout the training process did follow an ideal Hohmann transfer, although the optimizer is not confident in that solution as more exploration is required. Therefore, it is possible to utilize this formulation for solving individual design problems offline, effectively using Monte-Carlo simulations to exhaustively search the solution space. This, of course, cannot be generalize for solving novel problems using the pre-trained model.

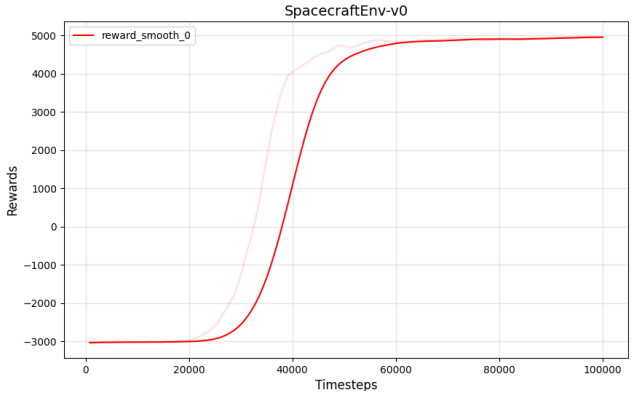

Fig. 1. Learning Curve for Sparse Reward function

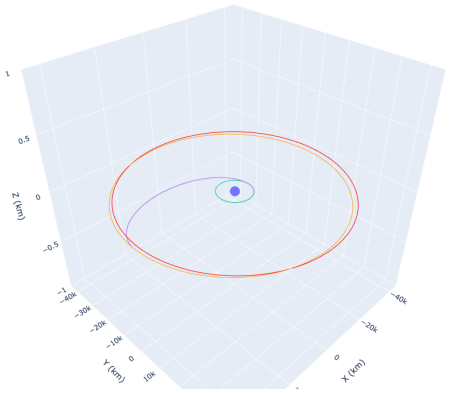

Fig. 2. Learned Trajectory for Sparse Reward Function

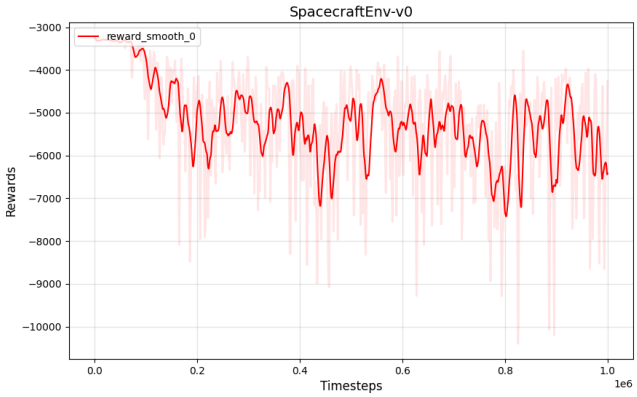

Fig. 3. Learning Curve for Sparse Reward Function with Large Action Space

C. Approach 3: Dense Linear Reward Function

When implementing the dense linear reward function with the PPO algorithm, the scaling of the reward based on proximity to the goal orbit inadvertently forces the transfer orbit trajectories to align with the semi-major axis (a) and eccentricity (e) of the goal orbit. This approach assumes that the transfer orbit should share similar orbital elements as the final orbit. However, in reality, this is not necessary. For instance, while the target orbit might be circular, the transfer orbit may need to be highly elliptical. Thus, rewarding the transfer orbit for being closer to the final orbit's parameters becomes counterproductive, leading to suboptimal behavior. This is demonstrated in Figure 5, where the transfer orbit trajectories clearly match the shape of the goal orbit, however they are poor transfer trajectories.

This misalignment is also reflected in the learning curve (4, which shows a noisy horizontal pattern with an average reward slightly below 0, indicating poor learning progress. The lack of improvement over time suggests that the agent struggles to find promising trajectories due to inappropriate intermediate rewards.

This outcome demonstrates that the dense reward structure is ineffective in this context. For a dense reward function to succeed, it must be designed to guide intermediate steps based on meaningful transfer orbit criteria rather than directly comparing transfer and goal orbit parameters. For instance, the reward could instead focus on orbital energy changes or incremental progress towards an efficient Hohmann transfer.

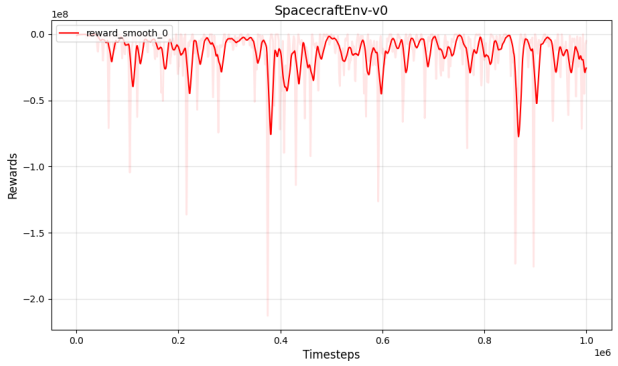

Fig. 4. Learning Curve for Dense Linear Reward Function

D. Approach 4: Hohmann Reference Reward Function

The final approach considered to improve learning performance, was to define a reward function with respect to a reference orbit. This was intended to help guide the RL agent to sample actions that closely align with an expected intermediate state. The results, however, did not show any considerable improvements in the learning process . It appears that the agent is continuing to explore the action space during the training period, rather than converging on the optimal policy. Perhaps providing stronger incentives to adhere to the reference trajectory would more adequately disabuse the agent

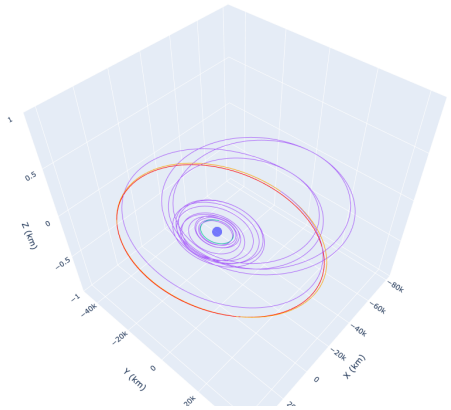

Fig. 5. Learned Trajectories for Dense Linear Reward Function

form deviating from the reference, although this imposition may result in suboptimal policies at convergence.

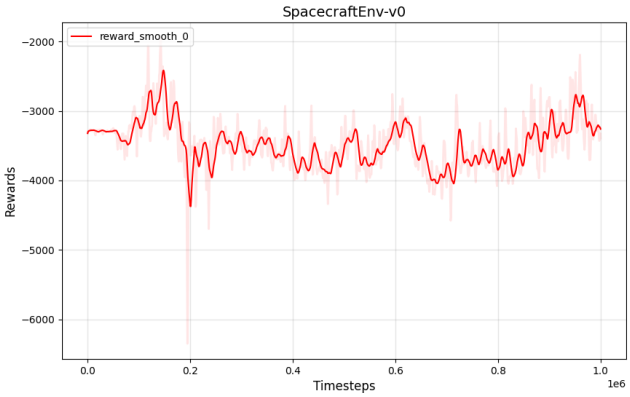

Fig. 6. Learning Curve using Hohmann Transfer Reference Trajectory

VI. Conclusions and Future Work

This project explored the application of reinforcement learning (RL) techniques to the problem of generating transfer trajectories between specified orbits while satisfying mission-specific objectives. Several approaches were considered, each with its own formulation of the reward function and action space.

The first approach utilized a sparse reward function and a reduced action space, resulting in successful learning of optimal transfer trajectories resembling Hohmann transfers. However, when the action space was expanded in the second approach, the learning performance deteriorated, indicating the challenges of dealing with larger action spaces.

In the third approach, a dense linear reward function was introduced to guide the agent towards orbits resembling the final goal orbit. However, this approach struggled to converge due to the misalignment between the reward structure and the desired trajectory characteristics.

Finally, the fourth approach introduced a reward function based on a reference trajectory, aiming to guide the agent towards trajectories resembling a Hohmann transfer. While

this approach showed some promise, further improvements are necessary to ensure effective learning.

Overall, this project highlighted the need for careful problem design when using reinforcement learning. Future work could focus on refining the reward functions to better guide the learning process, as well as exploring alternative RL algorithms or architectures that may be better suited to the problem domain. A continuous definition of the action space may have been a more appropriate representation rather than a finite discretization over the interval. Extending the application to more complex orbit transfers is also an avenue for further investigation.

Additionally, incorporating domain-specific knowledge or constraints into the RL framework could further enhance the performance and applicability of the proposed approach. There are several reinforcement learning techniques that could enhance performance. An autoregressive model structure could improve decision-making by embedding the wait action first and using its representation to inform the thrust action. A warmup learning rate schedule, starting low and gradually increasing, would prevent overfitting to initial samples and help avoid local minima, which is useful given the sensitive dynamics of orbital transfers. Additionally, physics-informed neural networks could embed orbital mechanics constraints directly into the learning process, ensuring that PPO policies stay within feasible physical bounds. Finally, a hierarchical RL approach could split orbital transfers into sub-problems with sub-goals, allowing tailored sub-policies to handle orbit insertion, trajectory correction, and final transfer, potentially improving learning efficiency and performance. These strategies collectively offer promising directions to refine RL applications for orbital dynamics.

All together, these could help expand the RL approach to more complex applications, such as including more complex dynamics like circular restricted three-body (CR3B) or n-body dynamics. Multi-agent frameworks could additionally be explored for more coordinated trajectory planning in multi-spacecraft missions.

REFERENCES

[1] C. J. Sullivan, N. Bosanac, R. L. Anderson, A. K. Mashiku and J. R. Stuart, "Exploring Transfers between Earth-Moon Halo Orbits via Multi-Objective Reinforcement Learning," 2021 IEEE Aerospace Conference (50100), Big Sky, MT, USA, 2021, pp. 1-13, doi: 10.1109/AERO50100.2021.9438267. *This paper explores the use of multi-objective reinforcement learning in planning trajectories between Earth and Moon, this paper was the basis of the project.*

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017. arXiv: 1707.06347v2 *This paper is the paper that introduced the PPO Algorithm, being a key resource in understanding the implementation for the project.*

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, "OpenAI Gym", 2016. arXiv:1606.01540 *This project provides a common framework for defining reinforcement leaning environments, useful for integrating with off the self solvers.*

[4] N. Barhate, "Minimal PyTorch Implementation of Proximal Policy Optimization," 2021, https://github.com/nikhilbarhate99/PPO-PyTorch.*This project provided an implementation of PPO using PyTorch which utilized the Gym API to train with our custom environment. Provided good insight into the various aspects of the Actor-Critic architecture.*

[5] N. Bosanac, "Leveraging Natural Dynamical Structures to Explore Multi-Body Systems," Ph.D. thesis, 2016. *This thesis offered insights into dynamical structures for trajectory design, although led to project downscoping to two-body dynamics.*

Member Tasks and Contributions

Carlos Chavez: MDP Problem formulation, reward shaping analysis, PPO tuning.
Ashwin Raju: MDP Problem formulation, integration of orbital dynamics, reward shaping analysis.
Yusif Razzaq: MDP Problem formulation, Development and integration of OpenAI gym platform and PyTorch-PPO. Contributed to sections II.A-B, IV.A-C, V.A-B
- The authors grant permission for this report to be posted publicly.