# Thrust Vectored Rocket Landing Integrated Guidance and Control with Proximal Policy Optimization

Gabriel de Almeida Souza[1] and Octávio Mathias Silva[2]

***Abstract*— This paper presents a 3 Degrees-of-Freedom rocket landing model environment, controlled by an agent trained with Proximal Policy Optimization (PPO) reinforcement learning algorithm. The objectives of this work are to model the dynamics of a rocket and its environment, convert into a simulated space adequate to reinforcement learning, and asses PPO training results. The proposed model is a 3-DoF longitudinal rocket with mass-varying properties, landing gear, and stochastic wind disturbances. The environment is modeled with an observation space composed of kinematic and contact properties, despite variation in mass and environmental perturbation. The action space is composed of three elements: main thruster effort, nozzle angle, and side thruster effort. The reward computation is based on state, fuel consumption, action transitions and termination status. It is known that simple control techniques can not prove stability for such complex problems. Reinforcement Learning (RL) is chosen to tackle the complexity of the problem and PPO for its theoretical training stability and continuous space treatment, in both observation and action space. Training and policy deployment assessments are presented to verify the algorithm efficacy and controllability of the proposed problem.**

## I. INTRODUCTION

Reusable launch vehicle technology is one of the latest great efforts tackled by the space industry. It has enormous potential to make space services more accessible, democratizing and enabling this market [1]. Specifically, reusable vertical landing rockets is yet a maturing technology, gaining traction [2], with demonstrated successful landings in the hundreds, as of 2022. Further developments can profoundly impact space economics [3]. The potential of this area has attracted attention from many engineering and computer science branches, such as guidance and control technology and machine learning applications.

Vertical powered landing has been deemed fundamental, because it has greater landing accuracy than the alternative method: passive parachute landing in distant sea areas. Regarding powered landing, the common control hardware design includes thrust vector actuators and cold gas lateral torque thrusters [4]. Research is not limited to hardware, but also includes designing guidance and control algorithms with new approaches potentially to be deployable under a real rocket and embedded computers.

This work was partially supported by CAPES.

[1]Gabriel de Almeida Souza is a graduate student in Electronic and Computer Engineering - Systems and Control, Aeronautics Institute of Technology, Sao Jose dos Campos, Brazil gabriel.gasouza@ga.ita.br

[2]Octavio Mathias Silva is a graduate student in Aerospace Engineering, Aeronautics Institute of Technology, Sao Jose dos Campos, Brazil octavio.silva@ga.ita.br

Aerospace systems have complex dynamic models and environment interactions, which lead to model imperfections and uncertainty when designing guidance and control algorithms. Critical maneuvers, such as powered landing, need advanced techniques to be optimally and robustly performed, making techniques, such as direct frequency domain design and pole allocation, not enough [5]. Simple linear control techniques are not sufficient.

Research in classical control theory include, among other techniques, adaptive nonlinear, parameter-varying control, and Model Predictive Control (MPC) [6], [7], [8], [9]. The main research performance qualities are fuel consumption minimization, robustness, applicability and generating safe trajectories. Another interested branch is machine learning, more specifically Reinforcement Learning (RL). Hierarchical Reinforcement Learning and Twin Delayed Deep Deterministic Policy Gradient have been explored in recent works [10], [11], with promising results for the successful landing of rockets using closed-loop control with a RL agent. Ferrante [12] assesses some classical control techniques and RL algorithms in a 3 Degrees-of-Freedom (DoF) rocket landing domain, contrasting each method characteristics.

Differing from control theory, RL applies a policy learned through experience. RL interacts with its environment by observing states, collecting rewards, and applying actions, iteratively through multiple episodes. The information gathered in episode batches, in policy gradient methods, is used to train the actor and critic models. This explains why building a simulated environment is fundamental to train a RL agent, specially in an aerospace domain, because it is prohibitive to train under the real setting.

Proximal Policy Optimization (PPO) [13] is a state of the art Deep RL algorithm, whose novelty is a learning expression that has better policy training stability, using a heuristic learning function that in practice ensures monotonic improvement. It deals with continuous action and observation spaces by using universal approximators, neural networks, as actor and critic models. It also does not require an internal model representation, which makes it more robust to parametric, process, and observation noise. It in an on-policy learning method. Gaudet et al. [14], [15] present a comparison between MPC and RL methods, and have shown sample results from applying PPO to rocket landing 3-DoF and 6-DoF problems.

In this paper, the proposed set of objectives is: To specify mission boundary conditions; to model an one-stage mass-varying actuated rocket within a 3-DoF simulation; to design a proper RL environment; and to tune and apply the PPO RL

algorithm for simultaneous guidance and control.

This paper is structured as follows: Section II presents the rocket dynamic modelling, including aerodynamic loads and randomness modelling, and landing criteria. Section III introduces environment modelling to better serve reinforcement learning, and PPO specific training information. Section IV presents training evolution, and deployment results. Finally, Section V presents the conclusions.

## II. ROCKET MODEL AND MISSION

The mission is defined as a Descent and Landing (DL) problem, starting close to Earth's surface, then $g$ is considered constant. The environmental domain is composed of a rocket with actuators, a fixed landing barge and setpoint, and the atmosphere. The domain is bidimensional, and the rocket is constrained to a longitudinal 3-DoF movement.

The proposed rocket contains an one-stage liquid engine with thrust vector control, cold gas side thrusters for pitch control, and landing gear to contact the barge. It is an underactuated system, yet all pose states $r$ in the domain are reachable within the fuel capacity $m_{fuel}$ constraint. The atmosphere exerts forces in the rocket through wind and drag. The barge is modeled as a fixed immovable object.

Figure 1 represents the domain coordinate system. The rocket pose is defined as $r = [x\ h\ \theta]^T$, with $x$ and $h$ being horizontal and vertical coordinates, and $\theta$ the angle between h-axis and the longitudinal axis of the rocket, or simply pitch. Nozzle deflection $\delta$, main thruster force $T_M$, right $T_R$ and left $T_L$ thrust forces, are the actuated variables. Objective coordinates $x_r$ and $h_r$ are located in the barge surface.

The rocket contains properties $m$ and $I$ as current mass and pitch moment of inertia. The rocket is subject to gravitational force $W = -mg$ and aerodynamic efforts $F_x$, $F_h$ in the horizontal and vertical planes, and moment $M$.

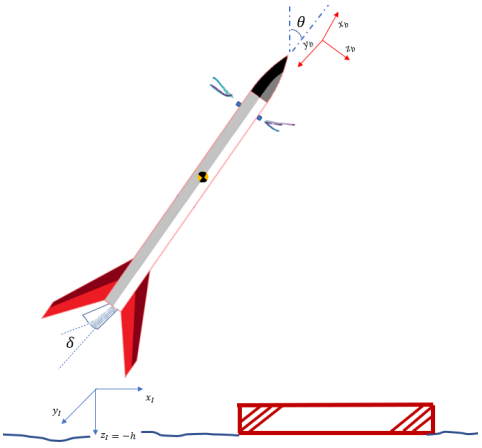



Fig. 1. Domain Coordinate System with Rocket and Objective

In the following subsections, the mathematical models for boundary conditions, dynamics, control restrictions and successful landing criteria are presented.

### A. Boundary Conditions and Success Criteria

$X^*$ denotes the vector with all time-varying properties. State space $X \in X^*$ is composed of the kinematic state vector, stacked with the mass, defined as $X = [m\ r^T\ \dot{r}^T]^T$, with $\dot{r} = [v_x\ v_h\ q]^T$.

State variables are initialized as described in Equation 1. Subindex 0 describes the initial instant.

$$\begin{bmatrix} m \\ x \\ h \\ \theta \\ v_x \\ v_h \\ q \end{bmatrix} = \begin{bmatrix} m_0 \\ \Omega(-x_0, +x_0) \\ h_0 \\ \Omega(-\theta_0, +\theta_0) \\ \Omega(-v_{x_0}, +v_{x_0}) \\ v_{h_0} + \Omega(-C_{vh}, +C_{vh}) \\ \Omega(-q_0, +q_0) \end{bmatrix} \tag{1}$$

Where $\Omega(a, b)$ describes a random variable with uniform distribution in the $(a, b)$ range, and $C_{vh}$ is a constant. This initial setting should prove useful to robustify the trained RL agent, given that variation in the DL mission starting condition is expected.

An episode terminal condition $\eta$ is given by a set of conditionals subject to an OR logical operation:

1) The rocket body contacts the barge;
2) The rocket moves beyond the $(x, h)$ valid domain coordinates;
3) Absolute pitch angle $|\theta|$ assumes a value greater than the safe $\theta_{limit}$ value;
4) Fuel ends at a high elevation;
5) The rocket landing gear contacts the barge.

Items 1-4 are clearly undesirable, and shall decrease the final reward function value. In item 5, if both legs are contacting the ground after time passing, the iteration will be counted as a success, and the final reward will be incremented. If inside a range of the setpoint, the reward will be much higher.

### B. Dynamics

Rocket Flight Dynamics is given by Equation 2. The first matrix to the right of the equation are system terms and the second matrix are terms including control variables $u$.

$$\begin{bmatrix} \dot{m} \\ \dot{x} \\ \dot{h} \\ \dot{\theta} \\ \dot{v_x} \\ \dot{v_h} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 \\ v_x \\ v_h \\ q \\ \frac{1}{m}[F_x] \\ \frac{1}{m}[F_h + W] \\ \frac{1}{I}[-\dot{m}qr_m^2 + M] \end{bmatrix} + \begin{bmatrix} f_m(T_M, T_R, T_L) \\ 0 \\ 0 \\ 0 \\ \frac{1}{m}[T_M cos(\delta + \theta) + (T_R + T_L)cos(\theta)] \\ \frac{1}{m}[T_M sin(\delta + \theta) + (T_R + T_L)sin(\theta)] \\ \frac{1}{I}[r_m T_M sin(\delta) + r_s(T_R + T_L)] \end{bmatrix} \tag{2}$$

Where $f_m$ is a function with non-positive range, dependent on thrust inputs. $r_m$ and $r_s$ are the longitudinal distance between the center of mass and thrust generation points in the main thruster and side thrusters, respectively. Aerodynamic efforts include drag $F_D$, and wind, which is modeled using a random uniformly distributed value proportional to $h$ with variations sampled from a gaussian distribution $\mathcal{N}(\mu(h), \sigma^2(h))$ in every $n$ time steps.

Actuator Dynamics are neglected because they are much faster than the rocket flight dynamics [16]. The control restriction are listed in Inequalities 3, and they include thrust magnitude, nozzle deflection limits, and available fuel. $T = [T_M \ T_S]^T$, with $T_S$ being the union of $T_L$ and $T_R$ ranges.

$$\begin{aligned} T_{min} \leq T \leq T_{max} \\ \delta_{min} \leq \delta \leq \delta_{max} \\ |\int \dot{m} dt| \leq m_{fuel} \end{aligned} \tag{3}$$

Contact dynamics are simple. If the rocket body or nozzle touch the barge, the episode is terminated. If the legs touch the barge, they act as rigid springs, with deformation proportional to contact velocity.

## III. Using Reinforcement Learning

In the previous section, the mission domain was defined, and the relevant dynamics were presented. In this section implementation details are presented, together with relevant RL specifics.

### A. Domain Simulation

A simulated domain, called the environment, is necessary to train the RL agent and to inspect results. OpenAI Gym [17] interface standard was used to design the domain and RL algorithm code. Box2D is a game physics engine, used to graphically render domain elements, such as the rocket, barge and other passive illustrative features. It is also used for collision dynamics and contact listening, and to integrate in a time step the equations of motion. The developed environment can be observed in Figure 2.

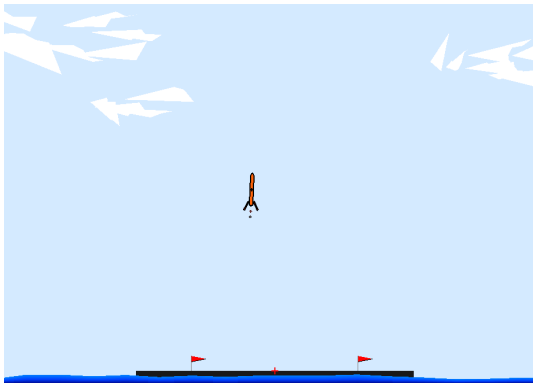


Fig. 2. Simulated Domain View

In order to learn how to actuate in this environment, RL agents adopt a Markov Decision Process approach [18]. The agent needs to do observations about relevant states $s \in X^*$, take actions $a \in u$, and receive reward signals $r$. The design is discussed in the following subsection.

### B. Observation, Action Models and Reward Shaping

RL does not require full-state feedback in order to converge to an useful policy in many problems. In this design, the environment returns the relative kinematic state vector $s$ to the agent, defined as $s = [\Delta x \ \Delta h \ \theta \ v_x \ v_h \ q \ \lambda_1 \ \lambda_2]^T$, with $(\Delta x, \Delta h) = (x - x_r, h - h_r)$, and $\lambda_1, \lambda_2 \in \{0, 1\}$ as binary variables indication landing gear contact. This observation vector was selected because it is simpler than full-state feedback, easier to train, and easier to deploy to a real embedded system.

In this problem observations are related directly to state, without any type of delay, noise or bias. This design choice was made to reduce the number of stochastic elements.

The action vector $a$ is defined as $a = [T_M \ T_S \ \Delta\delta]^T$, in which $\Delta\delta$ is the nozzle angle gradient in the time step. The environment will compute the action vector using the set of constraints presented in Inequalities 3, by clipping to maximum values larger values, and nullifying values below the minimum.

The reward function includes a shaping component $\Psi$, including kinematic state $s$, and termination status information $\eta$. Inverse reinforcement learning is used to reduce sparsity in reward signals, enabling and accelerating training. The reward also includes action $a$, and a term that punishes transitions to thruster activation, to encourage a "bang-bang" behaviour with less oscillation. The reward function is presented in Equation 4.

$$\begin{aligned} \Psi_k &= \alpha s_k + \kappa \eta_k - \Psi_{k-1} \\ R_k &= \Psi_k + \beta a_k + \zeta(H(|T|_k) \wedge \neg H(|T|_{k-1})) \end{aligned} \tag{4}$$

The subindex $k$ relates to the current iteration. $\Psi(0) = 0$. $\alpha$, $\beta$, $\kappa$ and $\zeta$ are coefficient vectors. $H$ is the step function, $\wedge$ and $\neg$ are logical AND and negation operators. The training hyperparameter discount rate $\gamma$ should play a role in making the agent more immediatist, which is also useful in generating quickly stabilizing trajectories.

### C. Training with PPO

PPO is trained using batches containing episodes which in turn contain time steps. A sequence of Monte Carlo episodes is generated, in order to stash information and train when enough data has been accumulated. As it is an on-policy method, actor and critic networks improve as training continues, as well as performance.

RL agents generally require environmental variability in training batches to obtain an actor and critic that are not overfitted. This is guaranteed by including randomness in initial conditions (Equation 1), and process noise as wind. Despite, the agent must also explore its observation-action mapping in order to improve the policy. This is done by sampling actions using a gaussian distribution with mean given by the deterministic policy $\pi_\theta(a_t|s_t)$ and a diagonal covariance matrix.

Learning and training posses other hyperparameters, such as the learning rate, PPO specific Kullback-Liebler divergence clip, timesteps per episode, per batch, and number of learning updates per iteration. [13] and [12] were consulted to start testing hyperparameters.

Normalized observation and action spaces are used to improve the actor and critic neural networks training, with backpropagation. The neural networks were implemented and trained using Python Torch with Adam optimizer with three layers, using ReLU activation function in the entry and intermediary, and linear in the last layer.

## IV. RESULTS

In this section a PPO agent is trained under a sequence of Monte Carlo episodes. Its training evolution is assessed by observing the mean cumulative reward and mean number of time steps per episode per iteration. Then the agent is deployed in the same environment to asses its efficacy and capacity to produce a robust policy. Control and state are analyzed from a sample.

Geometrical and dynamic domain proportions, PPO hyperparameters, and models can be observed in this repository: https://github.com/GabrieldeAS/PPO-lander.

### A. Policy Optimization

Figure 3 shows cases of reward and average duration of each episode per training iteration. It can be observed that the episodic reward grows with time, despite a poorer performance in some iterations, because of all the stochasticity. The same is inferred for episodic length, with a little decrease after iteration 350. It should be noted that episodic length has a sharper growth in the beginning, because the rocket first learns how to stabilize itself and extend its flight. Training was cut before converging, after 12 hours of training in a Intel i5 processor.

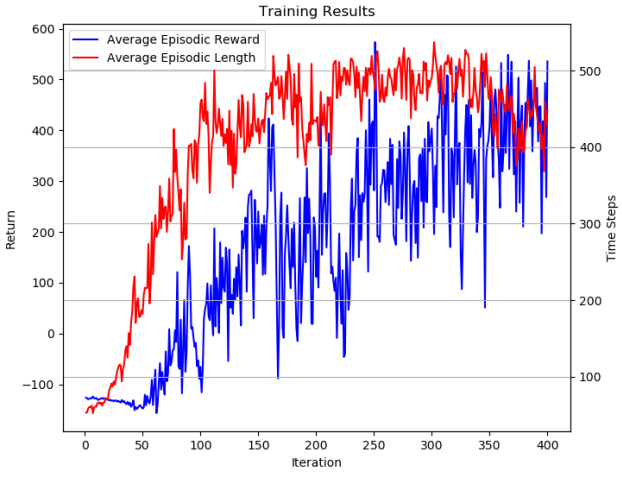



Fig. 3. Average length per episode and average reward per episode in each training batch iteration

### B. Policy Testing

Although training produces an actor and critic models, the actor model alone is sufficient for deployment with learning disabled. In this subsection, the obtained actor is assessed, using its deterministic policy.

Figure 4 presents the reward employing the trained actor for each iteration. The reward parameters were designed in a way that perfect landings receive much higher reward, and failures, usually below zero. Intermediary results are successful landings outside of the perfect landing range.

Another set of relevant metrics are successful landing and mean fuel consumption. The test setting obtained $80\%$ of successful landings and finished flight with $71\%$ of fuel with standard deviation of $3\%$. This performance showcases the behavior robustness learned by the policy, despite the variability in initial conditions and wind.

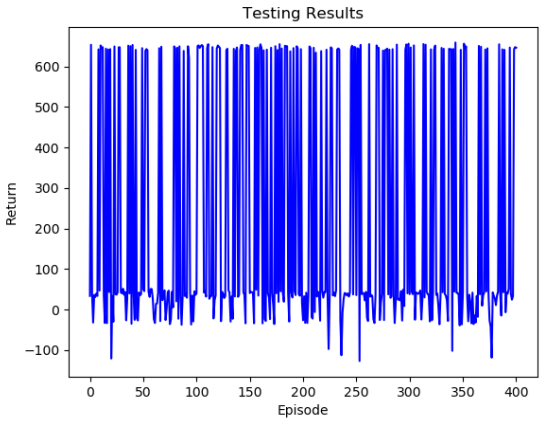



Fig. 4. Mean time steps per episode and Mean Reward per episode in deployment

A Monte Carlo sampling of trajectories can be observed in Figure 5. This collection has 100% successful landings. It can be noted that most trajectories were capable of approaching the setpoint, and finishing with a smaller error $\Delta x$.

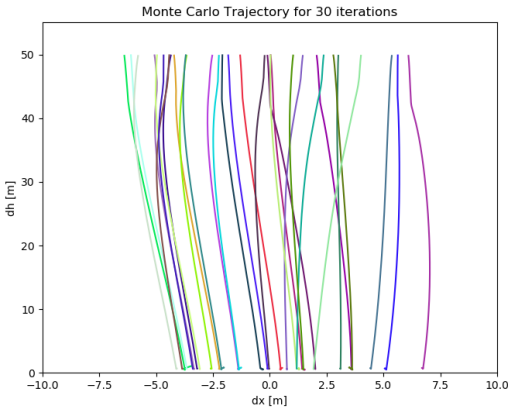



Fig. 5. Monte Carlo Obtained Trajectories

### C. Sample Flight Profile

A sample control and state profile is presented in this subsection. Figure 6 presents the selected sample trajectory, which ended with a successful landing. In this particular case a high initial $\theta$ caused interesting behavior.

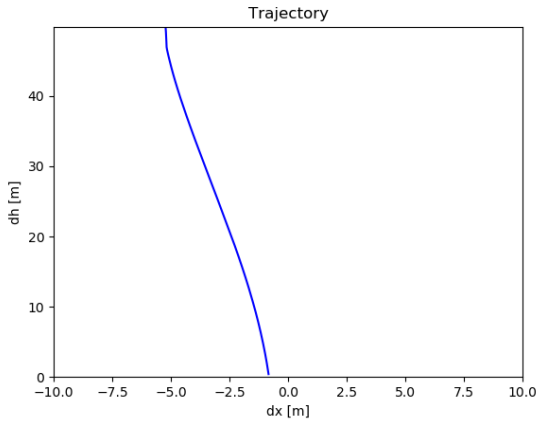
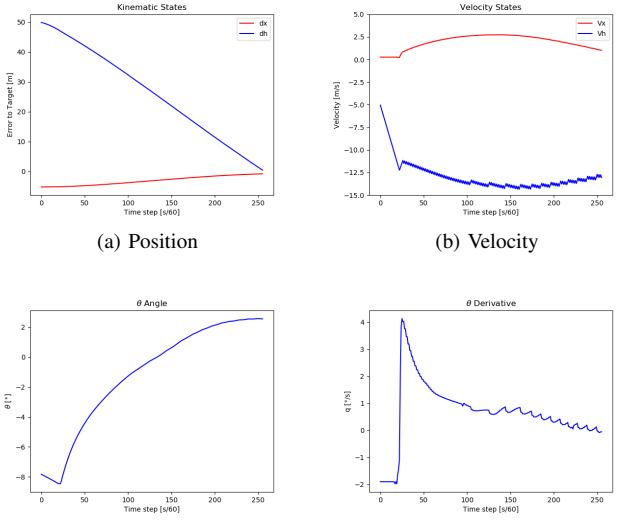



Fig. 6. Sample Control Profile and Trajectory

Figure 7 presents the rocket kinematic state evolution. The rocket position and velocity can be observed in (a) and (b), in which a smooth trajectory is observed, with small oscillations. Rocket pitch angle and derivative are observed in (c) and (d), with a continuous increase in $\theta$ with time, caused by an impulse in $q$ with continuation of positive values.




(a) Position (b) Velocity

(c) Pitch Angle (d) Pitch Derivative

Fig. 7. Kinematic States with Time

By inspecting, the controller behavior in Figure 8, it can be observed that side thrust $T_s$ was used briefly at the initial moments of the maneuver to correct pitch $\theta$, then never again. Nozzle angle $\delta$ is incremented with negative values in the beginning, to help control the pitch channel, then it was stabilized. Main thrust $T_m$ was continuously being turned on and off at minimal $T_{Mmin}$, to help control descent speed. This DL behaviour is satisfactory, although the ON/OFF transitions are yet too frequent and fast for a real system.

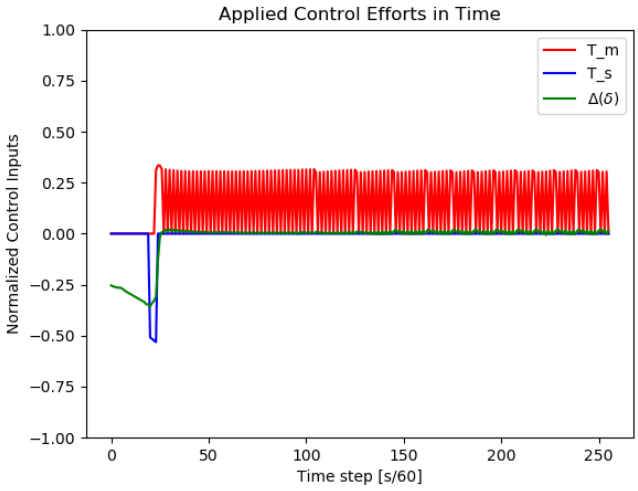



Fig. 8. Normalized Control Efforts

Figure 9 displays fuel consumption over time. It is mostly dependent in $T_m$, causing a linear steep to be generated, because of the propulsion pattern. It had enough fuel to land the rocket safely.

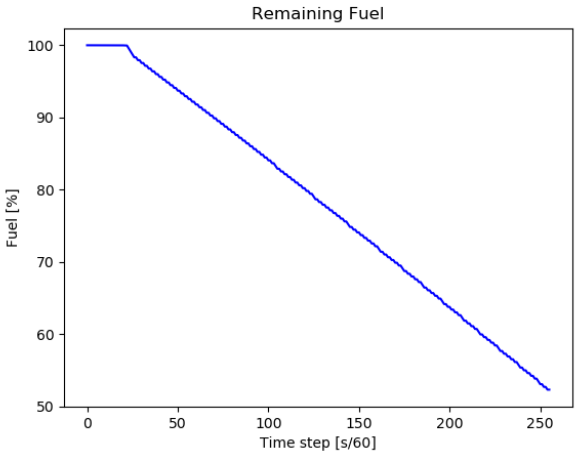



Fig. 9. Fuel Evolution

Observe that in the results observed in the previous subsection and the current one, the actor demonstrated good performance. Although it shows robust results, its edge case stability can not be exhaustively tested or proven. Still, RL is trustworthy in its training experience and is a great theoretical and practical step towards further advanced algorithms.

## V. CONCLUSIONS

This paper has presented the dynamic modelling of a landing stage rocket and environmental characteristics, and a PPO based control to successfully land the rocket. Although the physics were significantly simplified, the problem is still challenging for simple control strategies, and reinforcement learning has been shown to demonstrate good results.

Further improvements shall include faithful models for atmosphere and aerodynamics, and control actuator dynamics. Parametric noise is also relevant in developing a more vehicle robust policy. Observation uncertainty relating to the state space is also a necessary realistic assumption to approach real systems, which would require the incorporation of state estimation or further robustness. Assessing a set of reinforcement learning algorithms and control techniques is also interesting, to observe performance and real deployment potential.

## ACKNOWLEDGMENT

In this work, our RL agent and environment implementation borrowed from Ferrante's [12], which can be found at (https://github.com/arex18/rocket-lander).

We thank ITA for the academic formation.

We thank ITA Rocket design for the opportunity of working with Rocket Models. We also thank Bizu Space for the input curves and parameters used for the rocket model.

## REFERENCES

[1] Kelly Whealan George. 2019. The Economic Impacts of the Commercial Space Industry. Space Policy, Vol. 47, 181-186, ISSN 0265-9646. https://doi.org/10.1016/j.spacepol.2018.12.003.

[2] Harry W Jones. 2018. The Recent Large Reduction in Space Launch Cost. 48th International Conference on Environmental Systems. Albuquerque, New Mexico.

[3] Tománek, Robert and Hospodka, Jakub. 2018. Reusable Launch Space Systems. MAD - Magazine of Aviation Development. 6. 10-13. 10.14311/MAD.2018.02.02.

[4] Kai Dresia, Simon Jentzsch, Günther Waxenegger-Wilfing, Robson Dos Santos Hahn, Jan Deeken, Michael Oschwald, and Fabio Mota. 2021. Multidisciplinary Design Optimization of Reusable Launch Vehicles for Different Propellants and Objectives. Journal of Spacecraft and Rockets. Vol. 58, No. 4, July–August 2021

[5] Jhon M Hanson. 2000. Advanced guidance and control project for reusable launch vehicles. AIAA Guidance, Navigation and Control Conference and Exhibit 14-17 August 2000

[6] Maopeng Ran, Qing Wang, Delong Hou, Chaoyang Dong. 2014. Backstepping design of missile guidance and control based on adaptive fuzzy sliding mode control. Chinese Journal of Aeronautics, Volume 27, Issue 3. Pages 634-642, ISSN 1000-9361, https://doi.org/10.1016/j.cja.2014.04.007.

[7] Abhinav Kamath. 2021. Robust Thrust Vector Control for Precision Rocket-Landing. Master Thesis in Mechanical and Aerospace Engineering, Office of Graduate Studies, University of California.

[8] Jinbo Wang, Naigang Cui, and Changzhu Wei. Optimal Rocket Landing Guidance Using Convex Optimization and Model Predictive Control. Journal of Guidance, Control, and Dynamics 2019 42:5, 1078-1092

[9] Behcet Acikmese and Scott R. Ploen. 2007. Convex Programming Approach to Powered Descent Guidance for Mars Landing Journal of Guidance, Control, and Dynamics 2007 30:5, 1353-1366

[10] Mikulis-Borsoi, Francesco Alessandro Stefano, "Landing Throttleable Hybrid Rockets with Hierarchical Reinforcement Learning in a Simulated Environment" (2020). Honors Theses and Capstones. 521. https://scholars.unh.edu/honors/521

[11] Han Yuan, Yongzhi Zhao, Yu Mou, Xiaojun Wang. Leveraging Curriculum Reinforcement Learning for Rocket Powered Landing Guidance and Control

[12] Reuben Ferrante. 2017. A Robust Control Approach for Rocket Landing. Master Thesis in Artificial Intelligence, School of Informatics, University of Edinburgh.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. OpenAI. arXiv:1707.06347

[14] Brian Gaudet, Richard Linares, and Roberto Furfaro. 2018. Integrated Guidance and Control for Pinpoint Mars Landing Using Reinforcement Learning. Conference Paper.

[15] Brian Gaudet, Richard Linares, Roberto Furfaro. 2020. Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Landing. Advances in Space Research. Volume 65, Issue 7, Pages 1723-1741. 10.1016/j.asr.2019.12.030.

[16] A. Das, T. Garai, S. Mukhopadhyay and A. Patra, "Feedback linearization for a nonlinear skid-to-turn missile model," Proceedings of the IEEE INDICON 2004. First India Annual Conference, 2004., 2004, pp. 314-317, doi: 10.1109/INDICO.2004.1497762.

[17] Brockman, Greg and Cheung, Vicki and Pettersson, Ludwig and Schneider, Jonas and Schulman, John and Tang, Jie and Zaremba, Wojciech. 2016. OpenAI Gym. arxiv:1606.01540

[18] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.