POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Title of the thesis

Laurea Magistrale in Space Engineering - Ingegneria Spaziale

Author: Davide Iafrate

Advisor: Prof. Michéle Lavagna

Co-advisors: Dr. Robert Hinz, Eng. Andrea Brandonisio

Academic year: 2022-2023

1. Introduction

This thesis explores the possibility of using Reinforcement Learning (RL) techniques for planetary landing, specifically in the context of reusable first stages of launch vehicles.

1.1. Motivation

The landing problem is achieving a successful touchdown on a planetary body within a prescribed location and velocity, with a certain attitude and null terminal angular velocity. This problem has been rising in relevance in recent years to make space access more economically accessible and for future planetary exploration goals of space agencies. Several techniques have been developed over the years, with the state-of-the-art guidance and control (G&C) architectures having separate subsystems for Guidance and Control, which can lead to performance limitations. At first polynomial offline guidance methods were used, notably on the Apollo landers. Lossless convexification methods have then been developed [2] to optimize in real-time the 3DOF landing problem, where the lander is treated as a point mass. Successive convexification (SCVX) methods have been developed [6] to iteratively optimize convexified problems for guidance computation to generate 6DOF optimal open-loop trajectories. Robust control techniques have been used in the European space sector to add a guarantee of stability and controller performance. However, these methods suffer from performance limitations due to the nature of the architecture.

The present work focuses on developing an integrated Guidance and Control solution, assuming the navigation subsystem can give an accurate estimate of the lander's state.

In the past decade novel learning-based G&C architectures have shown promising results to synthesize an integrated G&C policy capable of improving on the State-of-the-art by merging the guidance and control systems.

Several studies have applied AI and reinforcement learning to spacecraft G&C problems, using neural networks to approximate controller parameters or to generate optimal solutions. Deep Reinforcement Learning revealed [3],[4] to be a flexible approach that maps the state of the system to a control action and has been chosen for maximum flexibility in modeling the nonlinear dynamics of a launcher's first-stage landing scenario on Earth. The existing works on this technique focused on applications to other planetary bodies, and none employ direct control of gimbaled thrusters, which significantly changes the behavior of the lander. The limitations of

previous approaches are in their long convergence times, which limits the ability to iterate quickly on the shape of the reward function.
This thesis assesses the maturity of Reinforcement Learning (RL) techniques for the development of integrated G&C policies that can achieve successful atmospheric landing by directly controlling the launcher's actuators.

1.2. Objectives

The research aims to improve the speed of design iteration by addressing the problem of long run times for RL algorithms. Additionally, the study aims to quantify the robustness of the control policy obtained from the RL algorithm against unmodeled dynamics and parameter uncertainty. To achieve these objectives, a validated 6DOF simulation environment is developed, expandable with more actuators, and a training software pipeline that enables fast iteration on the design of the integrated guidance and control policy is architectured. The first step was to develop the simulator and environment for the RL algorithm to learn the control policy, followed by the development of the training software pipeline to enable rapid testing of changes to the environment and the algorithm.

2. Problem Dynamics

2.1. 3DOF Dynamics

A simplified 3DOF (3 degrees of freedom) model of a rocket was first used at first. This model assumes that only the aerodynamic axial force A is considered, neglecting the normal force. It also assumes that the inertia I of the rocket is computed using an average value for the mass m and that gravity g is uniform.
The reference frame used for the 3DOF dynamics has the vertical axis as the y-axis, as shown in 1, unlike the 6DOF dynamics, which use the x-axis as the vertical one.
The dynamics of the system are described by Eq.1.

$$\begin{aligned} m\ddot{y} &= Tsin(\theta+\delta) - A\sin\theta - mg \\ m\ddot{x} &= Tcos(\theta+\delta) - A\cos\theta \\ I\dot{\omega} &= -T\sin\delta\,(x_T - x_{CoM}) \\ \dot{m} &= -\frac{T}{g_0 I_s} \end{aligned} \tag{1}$$

Overall, this simplified model provides a basic representation of the rocket's dynamics, which can be used as a starting point for further analysis.

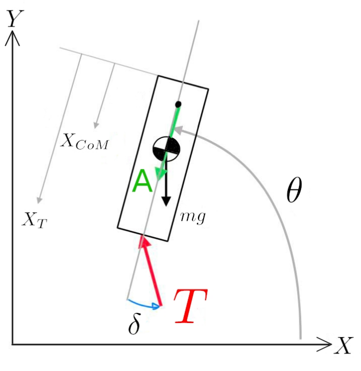

Figure 1: Reference frame used for the 3DOF dynamics. The forces acting on the launcher are the weight mg (applied in X_{CoM}), thrust T (gimbaled by an angle δ and applied in X_T) and the aerodynamic axial force A (applied at the center of pressure).

2.2. 6DOF Dynamics

A 6 degrees of freedom (6DOF) dynamics environment was then developed, for training an agent in a realistic scenario. This environment models both translational and rotational dynamics.

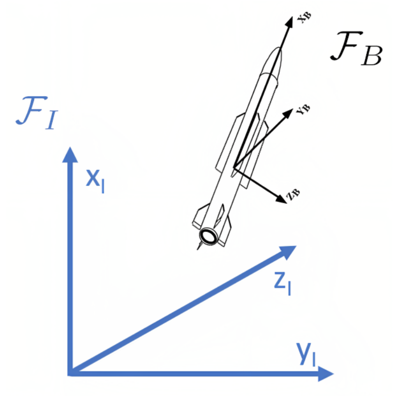

Figure 2: Inertial reference frame $\mathcal{F}_I$ (in blue) and body fixed reference frame $\mathcal{F}_B$ (in black).

The equations of motion and dynamics of the system are defined with respect to two reference frames: the inertial reference frame $\mathcal{F}_I$, fixed with the world and with its origin in the landing site, and a body-fixed reference frame $\mathcal{F}_B$ aligned with its x-axis along the longitudinal

axis of the vehicle.
The translational dynamics, shown in the equations 2, are defined in the $\mathcal{F}_I$ frame. They take into account the position $\vec{r}$ and velocity $\vec{v}$ vectors of the center of mass of the vehicle. The mass m is time-varying due to propellant consumption as modeled in 3, with the specific impulse of the engine being I_s and the standard gravitational constant at sea level g_0. The gravitational acceleration vector $\vec{g} = [-g_0, 0, 0]$.

$$\dot{\vec{r}}_I = \vec{v}_I$$
$$\dot{\vec{v}}_I = \frac{1}{m(t)}\vec{F}_I + \vec{g}_I \tag{2}$$
$$\dot{m} = -\frac{||\vec{T}_I||}{I_{sp}g_0} \tag{3}$$

In Eq.2 $\vec{F}_I$ is the sum of forces acting on the launcher, comprising:

- The control thrust force $\vec{T}_I$.
- The aerodynamic force $\vec{A}_I$, modeled using a *spherical aerodynamic model*. The atmosphere is modeled according to the ISA model.

The rotational dynamics is expressed in the body-fixed reference frame $\mathcal{F}_B$. It is computed using Euler's rigid body equation and the kinematics are parameterized using the quaternion representation in the scalar-first convention. They are shown in Eq.4.

$$\dot{\vec{q}} = \frac{1}{2}\vec{\Omega}\vec{q} \tag{4}$$
$$\dot{\vec{\omega}}_B = \vec{J}^{-1}(\vec{M}_B - \vec{\omega}_B \times \vec{J}\vec{\omega}_B)$$

2.3. Optimization Problem Formulation

The planetary landing problem involves reaching a final state with specific position and attitude while minimizing propellant consumption. This is modeled as an optimization problem with constraints on the system's dynamics and initial and final conditions. In a Reinforcement Learning setting, soft constraints are hinted through a reward function that penalizes their violation. The goal is to minimize propellant mass and errors in position, velocity, and attitude. The 3DOF formulation is similar to the 6DOF case but with a reduction in dimensionality of the vectors.

In mathematical terms the problem can be expressed as:

$$\begin{aligned}
&\underset{\vec{u}(t)}{\text{minimize}} && \Delta m = \int_{t_0}^{t_f} dm = m_0 - m_f && \text{(5a)}\\
&\text{subject to} && \dot{\vec{x}} = \vec{f}(\vec{x}, \vec{u}) && \text{(5b)}\\
&\text{with b.c.} && \vec{x}(t_0) = \vec{x}_0 && \text{(5c)}\\
& && \vec{r}(t_f) = \vec{r}_f && \text{(5d)}\\
& && \vec{v}(t_f) = \vec{v}_f && \text{(5e)}\\
& && \vec{q}(t_f) = \vec{q}_f && \text{(5f)}\\
& && \vec{\omega}(t_f) = \vec{\omega}_f && \text{(5g)}
\end{aligned}$$

The initial conditions sampled from a uniform distribution within a certain range, as shown in 4.3, and with the final conditions specified as Table 1.

$\vec{r}_f[m]$	$\vec{v}_f[m/s]$	$\vec{q}_f[-]$	$\vec{\omega}_f[rad/s]$
$\vec{0}$	$\vec{0}$	$[1, 0, 0, 0]$	$\vec{0}$

Table 1: Landing final conditions

3. Reinforcement Learning Algorithm

To solve the optimization problem a Reinforcement Learning algorithm is employed, specifically the *Proximal Policy Optimization.*

3.1. Proximal Policy Optimization (PPO) Algorithm

PPO is an on-policy algorithm that can be used for environments with either discrete or continuous action spaces. This algorithm has been selected due to its high wall-clock time efficiency during training, and being well suited for environments where it can be hard to assign a value to a state, such as one with sparse rewards. It works by directly shaping a policy $\pi(\vec{\theta})$ (hence being *on-policy*) to maximize the expected return over an episode. To have a more detailed overview of the algorithm the reader is referred to [5] and [1].

3.2. Neural Networks as function approximators

In the PPO algorithm, neural networks are used to estimate the *policy* function and the *value* function. The MultiLayer Perceptron (MLP) structure is employed, with two hidden layers of different sizes depending on the initial conditions: the first layer has more neurons in the 6DOF case, as it needs to learn a more complex policy. The network has a nonlinear activation function, which allows it to be a universal approximator. Two types of activation functions, hyperbolic tangent ($tanh$) and Rectified Linear Unit ($ReLU$), have been tested. The former solves the exploding gradient problem in back propagation and should allow for faster convergence. The network structure and activation functions are chosen based on performance and stability in the 3DOF scenario.

4. Environment development and software pipeline

Reinforcement learning involves an environment called a Markov Decision Process (MDP) or Partially Observable Markov Decision Process (POMDP), which includes possible states, actions, rewards, and transition functions. The reward function assigns a reward at each transition.

4.1. Environment structure

The OpenAI Gym environment is used to standardize the environment in which the rocket landing problem is tested. The implementation of the environment defines both the state transition function and the reward function, and is compliant with OpenAI Gym APIs. The environment interacts with the policy through standard APIs, allowing for the use of standard Reinforcement Learning frameworks for training. The dynamics equations are integrated using a variable-step RK45 ODE integrator from the Python package SciPy. The environment's $.step(\vec{a})$ method advances the simulation at a *fixed* timestep and checks for zero height using an event function.

4.2. Observation and action spaces

The observation space and state space of the environment in the thesis are continuous spaces that contain all possible values of the observations and system states. To improve convergence of the RL algorithms, it is good practice to bound and normalize the observation space within a certain range. The state space normalization vector is selected based on reasonable maximum values for each variable. The episode is terminated if any of the values of the state exceeds the state bounds or if certain early termination conditions are met, including reaching zero height, the upper or lower position bounds, or a time limit.
The action space refers to all possible actions that can be taken by an agent in a Reinforcement Learning environment. These actions are sampled from a policy $\pi(\vec{\theta_k})$, which is parametrized with a mean and a standard deviation. It is preferred for each element of the action to be bound between -1 and +1 to facilitate convergence of the algorithm. In the case of the study, the action vector comprises of thrust and gimbal angles in the Y and Z direction, which are limited to specific ranges. The denormalization process involves multiplying the action by the bounds vector.

4.3. Initial Conditions

The algorithm was tested using two sets of initial conditions: simplified and realistic. The simplified set had lower height, velocity, and velocity directed downwards, while the realistic set was sourced from historic flight data of the Falcon 9 and simulated landing on a downrange location such as a barge in the middle of the ocean or a downrange landing pad. The mean and range for each state of the initial conditions for both sets were reported in tables 2 and 3 for the 6DOF case. The same values, but with appropriate dimensionality reduction, are used for the 3DOF case. The sampling of different initial is a form of *domain randomization*, which is a technique used to obtain a more robust policy by preventing the RL algorithm from *overfitting* to a specific initial condition.

	Mean $\vec{\mu}_{\vec{x}_0}$	Range $\vec{\Delta}_{\vec{x}_0}$
$\vec{r}\,[m]$	$[500, 100, 100]$	$[50, 10, 10]$
$\vec{v}\,[m/s]$	$[-50, 0, 0]$	$[10, 10, 10]$
$\vec{q}\,[-]$	$[1, 0, 0, 0]$	$[0.1, 0.1, 0.1, 0.1]$
$\vec{\omega}\,[rad/s]$	$[0, 0, 0]$	$[0.1, 0.1, 0.1]$
$m\,[kg]$	$41e3$	$1e3$

Table 2: Mean and range of simplified initial conditions

4.4. Reward functions

The reward function in Reinforcement Learning maps current state and action taken to a scalar

	Mean $\vec{\mu}_{\vec{x}_0}$	Range $\vec{\Delta}_{\vec{x}_0}$
$\vec{r}\,[m]$	$[2000, -1600, 0]$	$[10, 200, 0]$
$\vec{v}\,[m/s]$	$[-90, 180, 0]$	$[30, 30, 0]$
$\vec{q}\,[-]$	$[0.866, 0, 0, -0.5]$	$[0.1, 0.1, 0.1, 0.1]$
$\vec{\omega}\,[rad/s]$	$[0, 0, 0]$	$[0.05, 0.05, 0.05]$
$m\,[kg]$	$41e3$	$1e3$

Table 3: Mean and range of realistic initial conditions

value that tells it how good or bad the current world state and action taken are. The choice of an appropriate reward function is crucial in producing a reasonably descriptive result. The reward function is developed and tested using iterations through various versions, with different coefficients and target values. Two different reward functions were tested, due to the inability of the algorithm to learn a successful policy in the case of realistic initial conditions using the first one.
The reward functions are made up of the weighted sum of several terms that encourage different behaviors. Both reward functions contain attitude bounds penalty reward terms and negative reward terms for thruster usage to reward the agent for minimizing fuel consumption. Also in all of them a high terminal reward term rewards the agent for achieving landing.
For the simplified initial conditions a two-phase training approach was used: in the first phase a target velocity reward function, developed in [4] is used to teach the agent to land in the landing location; In the successive training phase an annealed reward function is used, where only the terminal bonus reward and attitude bounds rewards are provided.
For the more realistic initial conditions a target acceleration reward function was used in place of the target velocity one, with the target acceleration being the solution of a simplified 3DOF problem minimizing the integral of the square of the acceleration of the launcher. This reward function teached the agent to make a successful landing, with the thruster negative reward term encouraging minimization of fuel consumption.

4.5. Software pipeline for rapid iteration

The PPO algorithm used for this study was from Stable Baselines3, a validated baseline. Training runs were initially executed on local machines, but when the computational burden became too high, they were scaled up and run in parallel on virtual machines through Google Cloud. The software was run in a Conda Python environment to ensure reproducibility and increase code reusability and Docker images are available. The results of each training run were uploaded to the web visualization tool Wandb for easy analysis and comparison. The environment state was visualized using Pygame for the 3DOF case and a 3D OpenGL rendering engine for the 6DOF case.

5. Results

5.1. 3dof environment

The performance of the RL algorithm in a simplified 3DOF landing control problem is satisfactory. In the simplified initial conditions case, che agent maximizes the reward by using thrusters at minimal levels while gaining speed, then performing a high-thrust final burn as shown by Fig.3. ReLUs activation functions were tested and resulted in quicker convergence, but have convergence instability, while hyperbolic tangent activation functions are more stable. The RL algorithm converges to a robust policy with low terminal velocity and position errors.

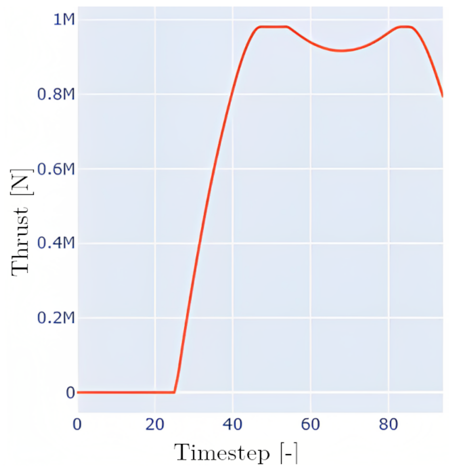

Figure 3: Thrust profile of a sample episode, showing the agent using a bang-bang profile to minimize gravity losses during the final burn.

In the realistic initial conditions case the RL algorithm successfully performs in the 3DOF landing control problem, first achieving successful landing and then minimizing the fuel consumption.

5.2. 6dof environment

Also for this case, two training runs are shown; the first run is for a set of simplified initial conditions, and the second is based on historic flight data of the Falcon 9.

Simplified initial conditions The results show that the algorithm successfully converges to a model with low propellant consumption and a low terminal velocity. The used propellant mass shown in Fig.4 also highlights quick convergence. A Monte Carlo analysis is done to compare the difference between the best policy obtained during the first phase and the policy obtained after the second phase of training. The second policy uses about 30% less propellant and has a lower average velocity error but a higher position error.

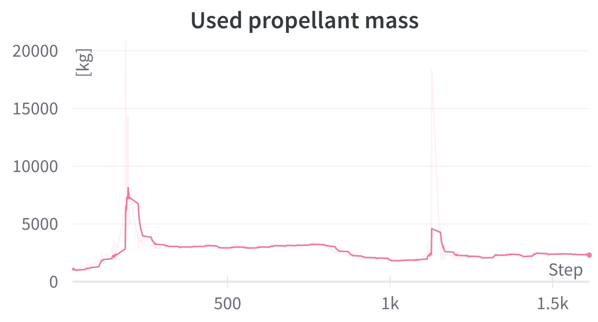

Figure 4: The trend in used propellant mass for the simplified initial conditions case. It is clearly visible the two-phase training process with a minimization of fuel consumption after the 1k steps mark.

The 6DOF environment has been validated by using a Simulink simulator validated through a Functional Engineering Simulator.

Realistic initial conditions With realistic initial conditions the *batch size*, measuring the number of transitions used per mini-batch in the optimizator, and the number of steps per roll-out, measuring the number of environment steps collected before performing policy optimization, had to be increased significantly to stabilize the policy updates and obtain convergence. Due to the highly oscillatory behavior of the performance metrics, which can be observed in Fig.5, a periodic evaluation of the policy is done, and the highest-performing (the one with highest mean

	RL controller (6DOF)	3DOF optimal solution	SCVX (6DOF)
$\lvert m_0 - m_f \rvert$	$4250\,kg$	$3545\,kg$	$7525\,kg$

Table 4: Propellant consumption comparison of the obtained policy, the 3DOF (point mass) optimal solution and 6DOF successive convexification (SCVX) approaches

reward) policy is saved. The convergence requires significant more episodes.

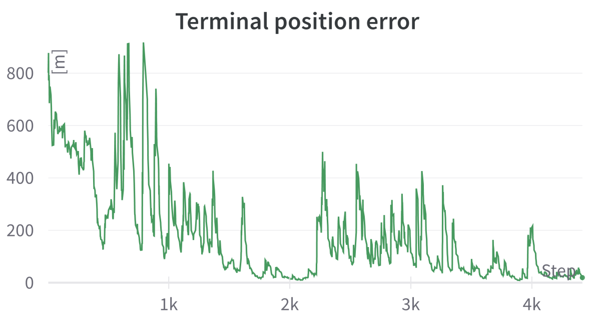

Figure 5: The trend in terminal position error shows convergence to low position errors but also high exploration, requiring to periodically evaluate the agent to save the best policy

A Monte Carlo analysis is carried out on to evaluate the terminal position and velocity errors. The RL solution is compared to a successive convexification MPC approach, and a 3DOF optimal solution obtained using a nonlinear optimizator, with the fuel consumption detailed in 4.

5.3. Sensitivity analysis

Finally an investigation of the robustness of the developed RL-based controller to unmodeled dynamics and disturbances is performed. The shift in position of the center of mass due to fuel and oxidizer consumption, as well as the flexural dynamics of the vehicle, have been modeled. The study also examines the effects of uncertainty in the inertia moments, real dynamics of the actuators, misalignment of the thrust and wind gusts. The results show that the controller is robust to the specified ranges of uncertainties and disturbances, as evidenced by comparable position and velocity errors and successful pinpoint landings. There were some outlier trajectories that exhibited uncontrolled behavior, as can be seen in the landing dispersion plot in Fig.6, which could be addressed by modifying the reward function or employing a landing mode controller.

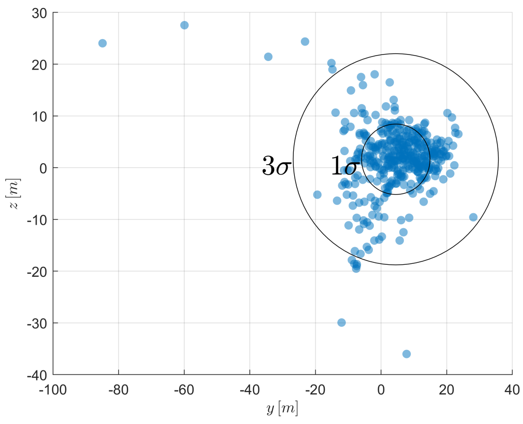

Figure 6: Dispersion plot with disturbances and unmodeled dynamics. Most of the simulations result in a successful landing within a $20m$ radius, however there are a few significant outliers.

6. Conclusions

This thesis presents the development and analysis of a Reinforcement Learning (RL) controller for the task of landing a reusable launcher's first stage, and quantifies its robustness to unmodeled dynamics and parametric uncertainties not present during training. Three main contributions are made: the application of a model-free RL algorithm to develop an integrated Guidance and Control (G&C) controller, the development of modular and easily expandable nonlinear 3DOF and 6DOF simulators, and the creation of an easy-to-use pipeline for cloud-accelerated RL training. The reward and activation functions, hyperparameters, and network structures are assessed and quantified in terms of their effects on convergence behavior and performance. The two different reward functions have different trade-offs, specializing better to different initial conditions. A two-phases approach in the reward function tested has proven promising in terms of performance, but improvements on its convergence behavior are needed to expand its generalization capabilities. The simulator has been validated, and policy robustness is achieved through domain randomization. Different topologies and activation functions have been tested for neural networks in terms of their convergence behavior and performance in terms of mean episodic reward. The activation function plays a key role in the convergence behavior, but once convergence is achieved, performance is similar. The topology of the network is also influential: a small network architecture with two hidden layers of 64 neurons each initially obtained good performance in a 3DOF scenario but was unable to learn well in a 6DOF scenario. Doubling the first layer improved learning ability, but further expanding the network resulted in poor performance. The policies obtained for both simplified and realistic initial conditions and environments perform satisfactorily and result in a successful landing. The simplified set of initial conditions with a two-phases training process leads to better performance with lower terminal velocity and position errors and a more aggressive control action. While errors are slightly higher in the 6DOF environment with realistic initial conditions, the use of a shaped terminal reward improves convergence. The policy is shown to be robust by having low errors and dispersion in the presence of different disturbances and uncertainty, however the presence of outliers needs to be addressed. The controller proves to be robust, fuel-efficient, and capable of achieving pinpoint landing, making RL a promising approach to solving the planetary landing problem in a novel way. Using neural networks allows the policy to be used in real-time on existing commercial hardware.
The study suggests several areas for future development in the use of Reinforcement Learning for spacecraft landing control. These include exploring the use of different algorithms, such as model-based approaches and imagination-augmented algorithms, as well as parallelization techniques for more efficient training. A critical role is also played by the optimization of reward functions, and the addition of actuators to the control system, as well as addressing outliers and improving policy explainability. Finally, it would be interesting to move to the use of reinforcement meta-learning to develop a single controller for the entire descent trajectory.

7. Acknowledgements

This research was carried out during an internship in Deimos Space. I personally thank them for the opportunity and the support given.

References

[1] Joshua Achiam. Spinning Up in Deep Reinforcement Learning. *None*, 2018.

[2] Behçet Açıkmeşe and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2):341–347, 2011.

[3] Roberto Furfaro, Andrea Scorsoglio, Richard Linares, and Mauro Massari. Adaptive generalized zem-zev feedback guidance for planetary landing via a deep reinforcement learning approach. *Acta Astronautica*, 171:156–171, 2020.

[4] Brian Gaudet, Richard Linares, and Roberto Furfaro. Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Powered Descent and Landing. *arXiv e-prints*, page arXiv:1810.08719, October 2018.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.

[6] Michael Szmuk and Behcet Acikmese. Successive convexification for 6-DoF mars rocket powered landing with free-final-time. In *2018 AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, jan 2018.