Indian Institute of Space Science and Technology

Department of Avionics

Course Project Task 1

Lunar Lander v2

Course: Reinforcement Learning in Practice

Submitted by: Team Core Crunchers

Semester: VII

Date: October 26, 2025

Team Members

Aaditya Raj Anand, SC22B082

Ankit Raj, SC22B087

Shreya Jhawar, SC22B115

Vineesh K Vinod, SC22B153

Contents

Chapter 1

Introduction

The Lunar Lander problem is a well-known environment in reinforcement learning that involves controlling a spacecraft to perform a safe and precise landing on a designated landing pad. The original environment, as described in OpenAI's Gymnasium documentation, models the lander under the influence of lunar gravity, with control provided through discrete engine actions. The agent is expected to learn a policy that minimizes the number of actions while ensuring stability and a safe touchdown.

Motivation

The Lunar Lander task provides a balanced challenge between discrete control and continuous dynamics, making it an excellent case study for understanding both the theoretical and practical aspects of reinforcement learning (RL). However, directly applying classical dynamic programming or Markov Decision Process (MDP) methods such as *value iteration* and *policy iteration* to this problem is computationally expensive due to the continuous nature of the state and action spaces.

Hence, our primary motivation in this project was to simplify and reformulate the Lunar Lander problem into a discrete MDP framework that can be solved analytically using Python on a standard computer. This approach allows us to bridge the gap between classical reinforcement learning theory and the more complex model-free approaches like Deep Q-Learning that we will explore later.

Overview of Work

We began the project by studying the original Lunar Lander environment in detail — its action space, observation space, reward structure, and termination conditions — as defined in OpenAI Gymnasium. The complete problem statement is presented in Chapter 2.

After thoroughly understanding the dynamics, we realized that directly modeling all 8 continuous state variables (position, velocity, angle, angular velocity, and leg contacts) would make classical MDP techniques computationally infeasible. Therefore, we decided to simplify the environment by discretizing and reducing the state space while retaining the essential physical behavior of the lander.

The reduction was done in the following way:

- We retained only the key state variables: the horizontal position (x), vertical position (y), orientation angle (θ), and the two leg contact indicators (left leg and right leg).
- The velocity components in both x and y directions were removed by assuming that, once a thruster is fired, the lander moves with constant velocity in that direction. This assumption significantly reduces computational complexity without distorting the qualitative control behavior of the system.

This simplification allows us to express the problem in a discrete state space with manageable dimensions and to formulate transition and reward matrices that can be solved using MDP algorithms.

Environment Design and Visualization

Before proceeding to the MDP formulation, we first designed and visualized the simplified lunar lander environment using **Desmos**, an online graphical tool. This visualization helped us geometrically map the coordinates of the lander, landing pad, and boundaries, ensuring that our assumptions about motion and orientation were consistent with the physical layout.

Each possible state and action combination was analyzed to understand the geometric transitions between states, which later formed the foundation for constructing the transition probability matrix.

Formulating the MDP Model

Once the environment and state representation were finalized, we formulated the system as a Markov Decision Process. The MDP is defined by the following components:

- **States:** All combinations of (x, y, θ, L, R), where L and R denote left and right leg contacts.
- **Actions:** Four discrete control inputs corresponding to firing the left, right, or main thruster, or doing nothing.
- **Transition Probabilities:** Probabilities of moving from one state to another given a particular action, determined based on deterministic geometric motion assumptions.
- **Rewards:** A numerical reward structure designed to encourage landing at the center with correct orientation and penalize excessive tilt or crash.

Using these components, we generated the **transition probability matrix** and **reward matrix** for the simplified environment. These matrices form the mathematical core of the MDP formulation.

MDP Solution and Simulation

After defining the MDP, we implemented **value iteration** algorithm in Python using Google Colab. The algorithms were tested to compute the optimal policy for the simplified Lunar Lander problem.

The Python implementation consists of:

- Initialization of the state and action spaces.
- Construction of transition and reward matrices.
- Iterative computation of value functions until convergence.
- Extraction of the optimal policy from the converged value function.

The complete implementation, along with code structure and logic flow, is discussed in detail in Chapter 4.

Report Organization

This report is organized into four chapters:

- **Chapter 2:** Describes the original Lunar Lander problem in detail, including its action space, observation space, and reward function as defined by OpenAI Gymnasium.
- **Chapter 3:** Explains the process of simplifying and reformulating the problem as a discrete MDP, along with assumptions and derivations of the transition and reward matrices.
- **Chapter 4:** Details the Python implementation of value iteration methods on Google Colab.
- **Chapter 5:** Presents simulation results, interpretation, and outlines future extensions such as model-free RL approaches.

Conclusion

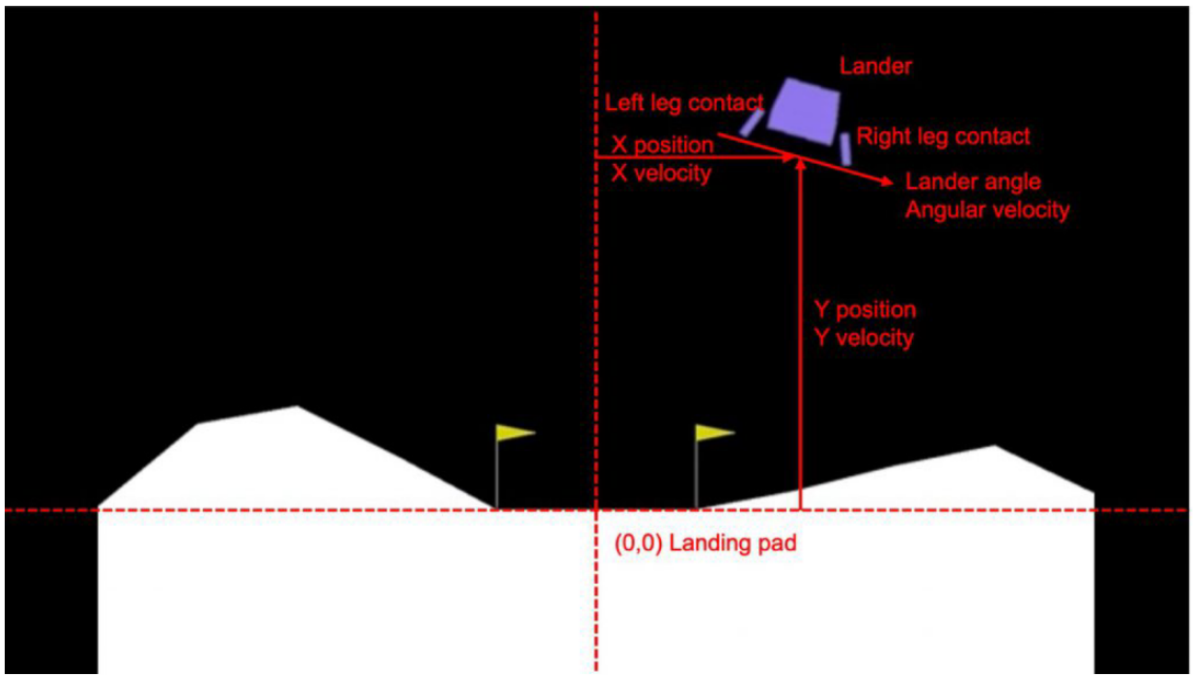
Through this project, we aimed to develop a deeper understanding of how a continuous control problem like the Lunar Lander can be reduced, modeled, and solved within the MDP framework. The insights gained here will serve as the foundation for implementing advanced model-free reinforcement learning algorithms in subsequent work.

Chapter 2

Problem Definition

The Lunar Lander problem is a classic sequential decision-making task where an agent must control a lunar lander to achieve a safe landing on a designated pad. The agent interacts with the environment at discrete time steps, observes its current state, selects an action, and receives a scalar reward as feedback.

The objective is to maximize the cumulative reward by learning a policy that maps observations to optimal actions. The environment used is the `LunarLander-v2` from `Gymnasium`, part of the Box2D environments. The landing pad is always located at coordinates $(0, 0)$.



2.1 Action Space

The environment provides four discrete actions that control the lander's thrusters:

- **0:** Do nothing.
- **1:** Fire left orientation engine.

- **2:** Fire main engine.
- **3:** Fire right orientation engine.

Note: According to Pontryagin's maximum principle, it is optimal to fire the engines at full throttle or not at all, which is why discrete actions are used.

2.2 Observation Space

The state is represented as an 8-dimensional vector:

1. Lander position along the x-axis.
2. Lander position along the y-axis.
3. Linear velocity along the x-axis.
4. Linear velocity along the y-axis.
5. Lander angle (orientation).
6. Angular velocity.
7. Boolean indicating if the left leg is in contact with the ground.
8. Boolean indicating if the right leg is in contact with the ground.

2.3 Rewards

Rewards are given at each step to encourage a safe and stable landing:

- Reward increases as the lander gets closer to the landing pad.
- Reward increases for slower velocities.
- Reward decreases if the lander is tilted (angle not horizontal).
- +10 points for each leg that contacts the ground.
- -0.03 points for each frame a side engine is firing.
- -0.3 points for each frame the main engine is firing.
- Additional -100 points for crashing or $+100$ points for landing safely.

An episode is considered solved if the total reward reaches at least 200 points.

2.4 Starting State

The lander starts at the top center of the viewport with a random initial force applied to its center of mass.

2.5 Episode Termination

The episode ends if:

- The lander crashes (body contacts the moon surface).
- The lander moves outside the viewport (x coordinate ¿ 1).
- The lander is not awake (a body that has come to rest and does not move or collide).

Chapter 3

Simplified MDP Formulation

This chapter presents a simplified discrete-state, discrete-action MDP model for Phase 1 of the Lunar Lander control policy. By discretizing states and actions, reducing dimensionality, and assuming known environment elements, the model preserves the key 2D lander dynamics while remaining computationally tractable for MDP-based control evaluation.

3.1 State and Action Spaces

3.1.1 State Definition

Each state of the system is represented as:

$$s = (x, y, \theta, l, r)$$

where:

- $x \in [-16, 16]$ — horizontal COM position (33 integer values)
- $y \in [-10, 22]$ — vertical COM position (33 integer values)
- $\theta \in \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$ — orientation (8 discrete angles)
- $l, r \in \{0, 1\}$ — binary leg contact indicators (left/right)

3.1.2 Action Set

$$\mathcal{A} = \{\text{noop},\ \text{left\_engine},\ \text{main\_engine},\ \text{right\_engine}\}$$

These correspond respectively to: no action, firing the left side thruster, the main engine, or the right side thruster.

3.2 Lander Geometry and Coordinate System

3.2.1 Reference Frame

The world is defined in a 2D Cartesian coordinate system (X, Y). The lander's center of mass (COM) at time t is:

$$\mathbf{p}_{\text{COM}}(t) = \big(X_{\text{COM}}(t),\, Y_{\text{COM}}(t)\big)$$

and its discrete orientation is:

$$\theta(t) \in \Theta = \{0,\ 45,\ 90,\ 135,\ 180,\ 225,\ 270,\ 315\}.$$

All positions lie on an integer grid:

$$X_{\text{COM}} \in [-16, 16], \quad Y_{\text{COM}} \in [-10, 22].$$

The full state vector at time t is:

$$s(t) = \big(X_{\text{COM}}(t),\ Y_{\text{COM}}(t),\ \theta(t),\ l(t),\ r(t)\big)$$

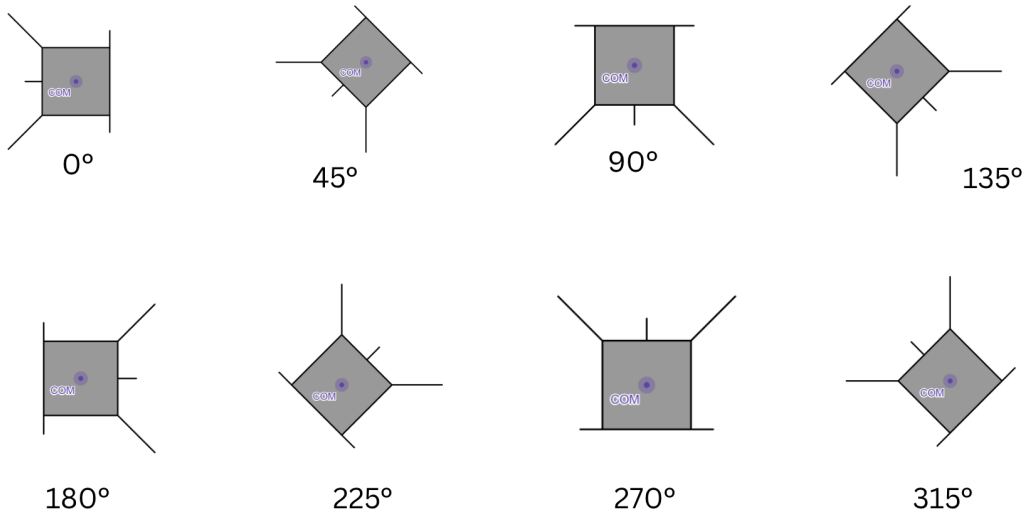


Figure 3.1: Possible orientations of the lander

3.2.2 Body-Frame Definition

At $\theta = 90$, the lander's geometry is defined relative to its COM:

$$\begin{aligned}
\text{Right thruster: } & (X_{\text{COM}} + 1,\ Y_{\text{COM}} + 1) \\
\text{Left thruster: } & (X_{\text{COM}} - 1,\ Y_{\text{COM}} + 1) \\
\text{Main engine: } & (X_{\text{COM}},\ Y_{\text{COM}} - 1) \\
\text{Right leg tip: } & (X_{\text{COM}} + 2,\ Y_{\text{COM}} - 2) \\
\text{Left leg tip: } & (X_{\text{COM}} - 2,\ Y_{\text{COM}} - 2)
\end{aligned}$$

At other discrete orientations, these coordinates are updated using precomputed offset tables that approximate rotation without continuous trigonometric computation. All geometric points are recomputed dynamically each timestep based on $(X_{\text{COM}}, Y_{\text{COM}}, \theta)$.

3.3 Landing Surface, Viewport, and Forbidden States

The visible world (viewport) is defined as:

$$P_1 = \{(X, Y) : X \in [-16, 16],\ Y \in [-10, 22]\}.$$

A simplified version of the landing surface is used to define the boundary for the MDP formulation. The surface is approximated using five line segments:

$$\begin{aligned} I_1 &: (-16, -4) \to (-10, -4), \\ I_2 &: (-10, -4) \to (-5, 0), \\ I_3 &: (-5, 0) \to (5, 0), \\ I_4 &: (5, 0) \to (12, 3), \\ I_5 &: (12, 3) \to (16, 0). \end{aligned}$$

The overall landing surface is the union:

$$\mathcal{S}_{\text{ground}} = \bigcup_{k=1}^{5} I_k.$$

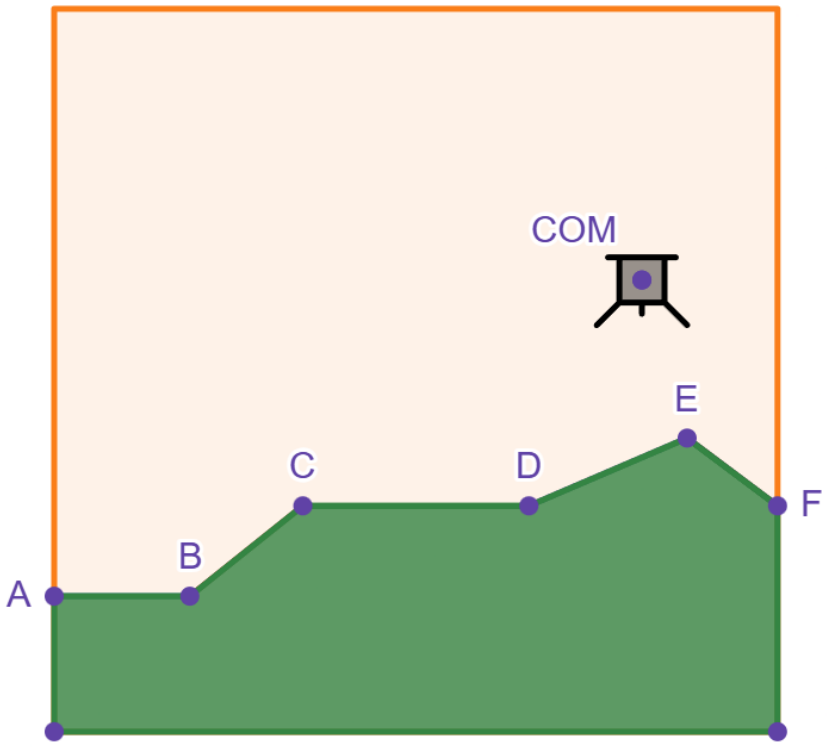


Figure 3.2: Landing Surface, Viewport, and Target Zone

3.3.1 Forbidden States

In the MDP formulation, certain positions are considered **forbidden states**. These are states where the center of mass (COM) of the lander cannot occupy because they lie below the ground surface, which is physically infeasible. Formally, a state

$$s = (X_{\text{COM}}, Y_{\text{COM}}, \theta, l, r)$$

is forbidden if

$$Y_{\text{COM}} < Y_{\text{ground}}(X_{\text{COM}}),$$

where $Y_{\text{ground}}(X_{\text{COM}})$ is the vertical position of the landing surface at the given horizontal coordinate.

For the simplified landing surface, $Y_{\text{ground}}(X_{\text{COM}})$ is defined piecewise as:

$$Y_{\text{ground}}(X_{\text{COM}}) = \begin{cases} -4, & -16 \le X_{\text{COM}} \le -9 \\ -3, & X_{\text{COM}} = -8 \\ -2, & X_{\text{COM}} = -7 \\ -1, & X_{\text{COM}} = -6 \\ 0, & -5 \le X_{\text{COM}} \le 7,\ X_{\text{COM}} = 15, 16 \\ 1, & 8 \le X_{\text{COM}} \le 9,\ X_{\text{COM}} = 14 \\ 2, & 10 \le X_{\text{COM}} \le 11,\ X_{\text{COM}} = 13 \\ 3, & X_{\text{COM}} = 12 \end{cases}$$

A forbidden state is then any state where the COM is below this surface:

$$s_{\text{forbidden}} = \{\, s \mid Y_{\text{COM}} < Y_{\text{ground}}(X_{\text{COM}}) \,\}.$$

Forbidden states are critical for the MDP because:

- They define boundaries that constrain the action space: the lander cannot move into these positions.
- They prevent the policy from selecting physically impossible maneuvers.

3.3.2 Target Landing Zone

The line segment I_3 $(-5, 0$ to $5, 0)$ represents the **target area** for landing. Successful landing is defined as the lander COM and both legs being within this region, respecting orientation and leg contact criteria.

3.4 Discrete Dynamics and Action Effects

3.4.1 Physical Constants (Discrete Approximation)

$$\text{Main thrust magnitude: } m = 2, \quad \text{Gravity: } g = 1.$$

Each step applies the following sequence:

$$\text{COM update (by action)} \longrightarrow \text{apply gravity } (Y \leftarrow Y - g) \longrightarrow \text{clamp to viewport.}$$

No-op (gravity-only):

$$\text{noop:} \quad \begin{cases} X(t+1) = X(t), \\ Y(t+1) = Y(t) - 1, \\ \theta(t+1) = \theta(t). \end{cases}$$

Main engine (net COM updates for each θ; θ unchanged):

θ	COM update $(X(t+1), Y(t+1))$
0°	$X(t+1) = X(t) + 2, \quad Y(t+1) = Y(t) - 1$
45°	$X(t+1) = X(t) + 1, \quad Y(t+1) = Y(t)$
315° $(=-45^\circ)$	$X(t+1) = X(t) + 1, \quad Y(t+1) = Y(t) - 2$
90°	$X(t+1) = X(t), \quad Y(t+1) = Y(t) + 1$
270° $(=-90^\circ)$	$X(t+1) = X(t), \quad Y(t+1) = Y(t) - 3$
135°	$X(t+1) = X(t) - 1, \quad Y(t+1) = Y(t)$
225° $(=-135^\circ)$	$X(t+1) = X(t) - 1, \quad Y(t+1) = Y(t) - 2$
180°	$X(t+1) = X(t) - 2, \quad Y(t+1) = Y(t) - 1$

The orientation remains unchanged:

$$\theta(t+1) = \theta(t) \quad \text{(main engine)}.$$

Left engine (side thruster): causes translation and a -45° change in θ.

θ	COM update and θ update
0°	$X(t+1) = X(t), \quad Y(t+1) = Y(t) - 3, \quad \theta(t+1) = \theta(t) - 45^\circ$
45°	$X(t+1) = X(t) + 1, \quad Y(t+1) = Y(t) - 2, \quad \theta(t+1) = \theta(t) - 45^\circ$
315°	$X(t+1) = X(t) - 1, \quad Y(t+1) = Y(t) - 2, \quad \theta(t+1) = \theta(t) - 45^\circ$
90°	$X(t+1) = X(t) + 2, \quad Y(t+1) = Y(t) - 1, \quad \theta(t+1) = \theta(t) - 45^\circ$
270°	$X(t+1) = X(t) - 2, \quad Y(t+1) = Y(t) - 1, \quad \theta(t+1) = \theta(t) - 45^\circ$
135°	$X(t+1) = X(t) + 1, \quad Y(t+1) = Y(t), \quad \theta(t+1) = \theta(t) - 45^\circ$
225°	$X(t+1) = X(t) - 1, \quad Y(t+1) = Y(t), \quad \theta(t+1) = \theta(t) - 45^\circ$
180°	$X(t+1) = X(t), \quad Y(t+1) = Y(t) + 1, \quad \theta(t+1) = \theta(t) - 45^\circ$

Right engine (side thruster): causes translation and a $+45^\circ$ change in θ.

θ	COM update and θ update
0°	$X(t+1) = X(t), \quad Y(t+1) = Y(t) + 1, \quad \theta(t+1) = \theta(t) + 45^\circ$
45°	$X(t+1) = X(t) - 1, \quad Y(t+1) = Y(t), \quad \theta(t+1) = \theta(t) + 45^\circ$
315°	$X(t+1) = X(t) + 1, \quad Y(t+1) = Y(t), \quad \theta(t+1) = \theta(t) + 45^\circ$
90°	$X(t+1) = X(t) - 2, \quad Y(t+1) = Y(t) - 1, \quad \theta(t+1) = \theta(t) + 45^\circ$
270°	$X(t+1) = X(t) + 2, \quad Y(t+1) = Y(t) - 1, \quad \theta(t+1) = \theta(t) + 45^\circ$
135°	$X(t+1) = X(t) - 1, \quad Y(t+1) = Y(t) - 2, \quad \theta(t+1) = \theta(t) + 45^\circ$
225°	$X(t+1) = X(t) + 1, \quad Y(t+1) = Y(t) - 2, \quad \theta(t+1) = \theta(t) + 45^\circ$
180°	$X(t+1) = X(t), \quad Y(t+1) = Y(t) - 3, \quad \theta(t+1) = \theta(t) + 45^\circ$

3.5 Leg Contact Detection and Terminal Condition

3.5.1 Contact Computation

At each step, the positions of the leg tips are computed from the current COM and orientation θ using a discrete rotation:

$$\begin{aligned}
\text{Left leg:} \quad & x_l(t+1) = x_l(t) + \cos\theta \cdot (x_l(t) - 2) - \sin\theta \cdot (y_l(t) - 2), \\
& y_l(t+1) = y_l(t) + \sin\theta \cdot (x_l(t) - 2) + \cos\theta \cdot (y_l(t) - 2). \\
\text{Right leg:} \quad & x_r(t+1) = x_r(t) + \cos\theta \cdot (x_r(t) + 2) - \sin\theta \cdot (y_r(t) - 2), \\
& y_r(t+1) = y_r(t) + \sin\theta \cdot (x_r(t) + 2) + \cos\theta \cdot (y_r(t) - 2).
\end{aligned}$$

A leg is considered in contact if its tip lies on or slightly below any ground segment I_k. These dynamically computed positions are used both for reward computation and terminal state detection.

3.5.2 Termination Conditions

The simulation terminates when:

$$Y_{\text{COM}}(t+1) \leq 0 \quad \text{or} \quad l(t+1) = 1 \quad \text{or} \quad r(t+1) = 1.$$

The terminal outcome is classified as:

- **Safe landing:** Both legs in contact ($l = r = 1$), within $\delta_x = 2$ of pad center ($X_{\text{pad}} = 0$), and upright ($|\theta - 90| \leq 10$). Reward: $+100$.
- **Crash:** Any other termination (single-leg contact, off-pad, or tilted). Reward: -100.

3.6 Reward Function for Simplified MDP

In the simplified MDP, the reward is designed to capture the key aspects of safe landing while remaining compatible with the discrete state and action representation. The reward at each timestep t is computed as follows:

3.6.1 Stepwise Reward

$$R(s(t), a(t), s(t+1)) = R_{\text{position}} + R_{\text{orientation}} + R_{\text{leg}} + R_{\text{engine}}$$

- **Position reward:** The closer the lander's COM is to the pad center ($X_{\text{pad}} = 0$), the higher the reward:

  $$R_{\text{position}} = -|X_{\text{COM}}(t+1) - X_{\text{pad}}|$$

  This encourages horizontal alignment over the landing pad.

- **Orientation reward:** The reward penalizes deviation from upright ($\theta = 90$):

  $$R_{\text{orientation}} = -\left\lfloor \frac{|\theta(t+1) - 90|}{45} \right\rfloor$$

  This encourages discrete upright orientation.

- **Leg contact reward:** Each leg in contact with the ground gives a bonus:

  $$R_{\text{leg}} = 10 \cdot (l(t+1) + r(t+1))$$

- **Engine usage penalty:** To encourage minimal fuel usage:

$$R_{\text{engine}} = \begin{cases} -0.3, & \text{if main engine fired} \\ -0.03, & \text{if side engine fired} \\ 0, & \text{if no engine fired (noop)} \end{cases}$$

3.6.2 Terminal Reward

At episode termination, an additional reward is assigned based on landing outcome:

$$R_{\text{terminal}} = \begin{cases} +100, & \text{safe landing: } l = r = 1,\ |X_{\text{COM}}| \leq 2,\ |\theta - 90| \leq 10 \\ -100, & \text{crash: any other termination} \end{cases}$$

3.6.3 Summary

The simplified reward function retains the essential incentives of the original problem:

- Encourages horizontal alignment over the landing pad.
- Rewards upright orientation.
- Rewards leg contact for successful touchdown.
- Penalizes excessive engine usage.
- Assigns strong terminal rewards for success or failure.

This formulation ensures that value or policy iteration in the discrete MDP remains computationally tractable while capturing the critical control trade-offs for a safe landing.

3.6.4 Implementation Notes

- Geometry recomputed every frame from (x, y, θ, l, r) without explicit rotation matrices.
- Orientation wrap-around handled mod 360.
- Optional small random disturbances (± 1 in x or y) can simulate environmental uncertainty.
- All coordinates clamped to the viewport: $X \in [-16, 16]$, $Y \in [-10, 22]$.

3.7 Reasoning for Simplification

The model intentionally omits velocities, continuous dynamics, and fine angular resolution to ensure computational feasibility:

- Continuous (x, y, θ) variables are discretized.
- Deterministic updates replace integration of continuous motion equations.
- Binary contact indicators simplify landing logic.

This reduced representation captures essential control trade-offs — balancing thrust, attitude, and safe touchdown — while enabling value or policy iteration within manageable computational bounds.

3.8 Summary

The proposed discrete MDP formulation provides a compact yet expressive approximation of the Lunar Lander problem. It enables optimal policy derivation through standard dynamic programming while preserving the key physics of thrust-driven descent, rotation, and landing safety. This forms the foundation for later phases involving stochasticity, velocity states, and continuous control refinements.

Chapter 4

References

1. OpenAI Gym Documentation: https://www.gymlibrary.dev/environments/box2d/lunar_lander/