Task 2 Report: Development of a Full Python-Based Lunar Lander Simulator for Reinforcement Learning

Team Name: Core Crunchers

Aaditya Raj Anand (SC22B082)
Shreya Jhawar (SC22B115)
Ankit Raj (SC22B087)
Vineesh K Vinod (SC22B153)

December 3, 2025

1 Introduction

This document presents a complete problem definition and mathematical modeling of a two-dimensional Lunar Lander system used for reinforcement learning (RL) based control and optimization. The goal is to construct a physically consistent yet computationally tractable simulator of a lunar landing vehicle, define its state and observation spaces, establish landing and crash conditions, and derive the governing equations of motion.

The environment is modeled as a Markov Decision Process (MDP) with continuous states and discrete control inputs. This simulator enables trajectory generation under random or policy-driven thruster firings, supporting RL algorithm training and evaluation.

2 2D World Definition

The world is defined in a 2D Cartesian coordinate system:

$$(X, Y) \in \mathbb{R}^2.$$

The lander's center of mass (COM) at time t is represented as:

$$\mathbf{p}_{\text{COM}}(t) = \begin{bmatrix} X_{\text{COM}}(t) \\ Y_{\text{COM}}(t) \end{bmatrix}.$$

The visible world (viewport) is defined as:

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

Gravity acts downward with acceleration g.

3 Lander Geometry

The lander is modeled as a rigid body with a square-like central chassis and two legs.

Angle Convention

The orientation angle θ is measured from the positive X-axis. The lander is *upright* when

$$\theta = 90^\circ = \frac{\pi}{2},$$

and positive rotation corresponds to anticlockwise motion.

Coordinates are defined relative to the COM when the lander is vertical:

$$\theta = 90^\circ = \frac{\pi}{2}.$$

In this configuration the following body-frame reference points exist:

$$\begin{aligned} \text{Right thruster: } & (+1, +1), \\ \text{Left thruster: } & (-1, +1), \\ \text{Main engine: } & (0, -1), \\ \text{Right leg tip: } & (+2, -2), \\ \text{Left leg tip: } & (-2, -2). \end{aligned}$$

These vectors will be rotated according to the current attitude θ.

3.1 Rotation Mapping

Any body-frame point $(x_{\text{loc}}, y_{\text{loc}})$ transforms to world coordinates as:

$$\boxed{\begin{aligned} x_w &= X_{\text{COM}} + \sin\theta \cdot x_{\text{loc}} + \cos\theta \cdot y_{\text{loc}}, \\ y_w &= Y_{\text{COM}} - \cos\theta \cdot x_{\text{loc}} + \sin\theta \cdot y_{\text{loc}}. \end{aligned}}$$

3.2 Leg Tip Coordinates

Using the above rotation, the leg tips become:

$$\begin{aligned} X_{R\ell eg} &= X_{\text{COM}} + 2\sin\theta - 2\cos\theta, \\ Y_{R\ell eg} &= Y_{\text{COM}} - 2\cos\theta - 2\sin\theta, \end{aligned}$$

$$\begin{aligned} X_{L\ell eg} &= X_{\text{COM}} - 2\sin\theta - 2\cos\theta, \\ Y_{L\ell eg} &= Y_{\text{COM}} + 2\cos\theta - 2\sin\theta. \end{aligned}$$

4 Landing Surface and Forbidden States

The landing surface is defined by five line segments:

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

This surface is approximated by a piecewise ground height function $Y_{\text{ground}}(X)$.

A **forbidden state** is any state where the COM lies below the ground:

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

5 Forbidden States

Certain positions of the lander are physically infeasible because the center of mass (COM) cannot lie below the ground surface. A state is therefore considered *forbidden* if

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

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

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

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

A forbidden state is therefore any state for which the COM lies strictly below the piecewise ground surface:

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

Forbidden states play a critical role in the MDP because:

- They define boundaries that constrain the feasible region of the state space.
- They prevent the agent from selecting physically impossible actions that would place the COM below the terrain.

6 State Space and Observation Space

The simulator maintains the following continuous-valued state:

$$s = (X_{\text{COM}}, Y_{\text{COM}}, v_x, v_y, \theta, \omega, \ell, r),$$

corresponding to position, velocity, angle, angular velocity, and left/right leg contact flags.
where ℓ and r are binary contact indicators for the left and right legs respectively.
Thus the observation/state vector is:

$$\boxed{\mathbf{o} = [X_{\text{COM}}, Y_{\text{COM}}, v_x, v_y, \theta, \omega, \ell, r]\,.}$$

The state dimension is 8.

7 Action Space

Actions correspond to thruster commands:

$$a \in \{0, 1, 2, 3\},$$

where:

0 : No thrust,
1 : Left thruster (produces a rightward force F_{right}),
2 : Main engine,
3 : Right thruster (produces a leftward force F_{left}).

8 Force Model

Forces act at the COM for this simplified model.

8.1 Total Forces

At orientation angle θ:

$$\boxed{F_x = F_{\text{main}} \cos\theta + (F_{\text{left}} - F_{\text{right}}) \sin\theta}$$

$$\boxed{F_y = F_{\text{main}} \sin\theta + (-F_{\text{left}} + F_{\text{right}}) \cos\theta - mg}$$

Linear accelerations:

$$a_x = \frac{F_x}{m}, \qquad a_y = \frac{F_y}{m}.$$

9 Torque Model

Only the side thrusters produce torque, using a constant lever arm of 1 unit.

$$\tau_{\text{left}} = -F_{\text{left}}, \qquad \tau_{\text{right}} = +F_{\text{right}}.$$

Total torque:

$$\boxed{\tau_{\text{tot}} = F_{\text{right}} - F_{\text{left}}.}$$

Angular acceleration:

$$\alpha = \dot{\omega} = \frac{\tau_{\text{tot}}}{I}.$$

10 Continuous-Time Equations of Motion

$$\dot{X} = v_x, \qquad \dot{Y} = v_y,$$
$$\dot{v}_x = \frac{F_x}{m}, \qquad \dot{v}_y = \frac{F_y}{m},$$
$$\dot{\theta} = \omega, \qquad \dot{\omega} = \frac{F_{\text{right}} - F_{\text{left}}}{I}.$$

11 Discrete-Time Update Equations

Let Δt denote the simulation time step. Two common numerical integration methods may be used to update the lander state. Both are valid for this simulator, and the choice depends on the desired balance between physical accuracy and numerical stability.

11.1 Method 1: Kinematic (Constant-Acceleration) Update

This method assumes that the linear and angular accelerations remain constant over the interval $[t,\, t + \Delta t]$. It yields the exact closed-form solution of the continuous-time equations under that assumption.

Linear Motion

$$v_x^{t+1} = v_x^t + a_x \Delta t, \qquad v_y^{t+1} = v_y^t + a_y \Delta t,$$
$$X^{t+1} = X^t + v_x^t \Delta t + \tfrac{1}{2} a_x \Delta t^2, \qquad Y^{t+1} = Y^t + v_y^t \Delta t + \tfrac{1}{2} a_y \Delta t^2.$$

Angular Motion

Let $\alpha = \tau_{\text{tot}}/I$ denote the angular acceleration.

$$\omega^{t+1} = \omega^t + \alpha \Delta t,$$

$$\theta^{t+1} = \theta^t + \omega^t \Delta t + \tfrac{1}{2} \alpha \Delta t^2.$$

Advantages:

- Exact solution under constant acceleration.
- Physically intuitive (matches textbook kinematics).

Disadvantages:

- Less numerically stable for rapidly changing forces.
- Can accumulate energy errors over long simulations.

11.2 Method 2: Semi-Implicit Euler (Symplectic Euler)

This method first updates velocities from accelerations, then updates positions using the new velocities. It is widely used in physics engines and reinforcement learning simulators due to its robustness and stability.

Linear Motion

$$v_x^{t+1} = v_x^t + a_x \Delta t, \qquad v_y^{t+1} = v_y^t + a_y \Delta t,$$
$$X^{t+1} = X^t + v_x^{t+1} \Delta t, \qquad Y^{t+1} = Y^t + v_y^{t+1} \Delta t.$$

Angular Motion

$$\omega^{t+1} = \omega^t + \alpha \Delta t, \qquad \theta^{t+1} = \theta^t + \omega^{t+1} \Delta t.$$

Advantages:

- Superior numerical stability, especially for systems with rotation.
- Conserves energy more effectively in long simulations.
- Commonly used in game engines and RL environments.

Disadvantages:

- Slightly less physically exact over a single time step.

Recommendation

Either method may be used in this simulator. For physical interpretability, the kinematic method is appropriate. For long-term stability and reinforcement learning experiments involving many rollouts, the semi-implicit Euler method is generally preferred.

12 Termination Conditions

The simulation terminates when any of the following conditions hold at time $t+1$:

$$Y_{\mathrm{COM}}(t+1) \le 0 \quad \text{or} \quad \ell(t+1) = 1 \quad \text{or} \quad r(t+1) = 1.$$

Once the simulation terminates, the outcome is classified into one of two categories:

Safe Landing

A safe landing is declared if all of the following conditions are satisfied:

- both legs are in contact with the ground:

$$\ell = r = 1,$$

- the lander is horizontally close to the landing pad center $X_{\mathrm{pad}} = 0$:

$$|X_{\mathrm{COM}} - X_{\mathrm{pad}}| \le \delta_x = 2,$$

- the lander is upright:

$$|\theta - 90^\circ| \le 10^\circ.$$

Reward for a safe landing:

$$\boxed{+100}.$$

Crash

Any termination that does not meet all the above criteria is classified as a crash. This includes:

- single-leg contact,
- contact outside the landing pad region,
- excessive tilt at contact,
- touchdown at unsafe horizontal offset,
- hitting the ground with the body before both legs touch.

Reward for a crash:

$$\boxed{-100}.$$

13 Rewards

The environment issues a scalar reward at every simulation step. The total episode return is the sum of per-step rewards plus any terminal bonus/penalty applied at episode termination.

We use a shaped reward composed of several additive terms that encourage proximity to the pad, low speed, upright attitude, minimal control usage, and stable touchdown. The per-step reward at time t is defined as:

$$r_t = r_{\text{pos}}(s_t) + r_{\text{vel}}(s_t) + r_\theta(s_t) + r_{\ell r}(s_t) + r_{\text{ctrl}}(a_t),$$

and at termination an additional terminal reward r_{term} is added:

$$R_{\text{episode}} = \sum_{t=0}^{T-1} r_t \; + \; r_{\text{term}}.$$

Below we define each term.

Position (progress toward the pad)

Let the horizontal distance to the pad be

$$d_t \;=\; \big|X_{\text{COM}}(t) - x_{\text{pad}}\big|, \qquad x_{\text{pad}} = 0.$$

Define the *change* in horizontal distance between two successive timesteps:

$$\Delta d_t \;=\; d_t - d_{t-1}.$$

We reward improvement (moving closer to the pad) and penalize degradation (moving away) using only a differential term:

$$\boxed{r_{\text{pos}}(s_t) \;=\; -\,c_{\Delta x}\,\Delta d_t}$$

with $c_{\Delta x} > 0$.

Interpretation:

- If the lander moves **closer** to the pad: $d_t < d_{t-1}$, so $\Delta d_t < 0 \Rightarrow r_{\text{pos}} > 0$ (positive reward).
- If the lander moves **away** from the pad: $\Delta d_t > 0 \Rightarrow r_{\text{pos}} < 0$ (penalty).

For the first timestep of an episode, set $\Delta d_t = 0$ so no spurious reward is issued.

Velocity (progress toward slower motion)

Let the instantaneous speed be

$$v_t \;=\; \sqrt{v_x(t)^2 + v_y(t)^2}.$$

Define the speed change:

$$\Delta v_t \;=\; v_t - v_{t-1}.$$

We reward speed reduction (slowing down) and penalize increases (speeding up):

$$\boxed{r_{\text{vel}}(s_t) \;=\; -\,c_{\Delta v}\,\Delta v_t}$$

with $c_{\Delta v} > 0$.

Interpretation:

- If the lander **slows down**: $v_t < v_{t-1}$, so $\Delta v_t < 0 \Rightarrow r_{\text{vel}} > 0$.
- If the lander **speeds up**: $\Delta v_t > 0 \Rightarrow r_{\text{vel}} < 0$.

As before, set $\Delta v_0 = 0$ at episode start.

Tilt (encourage upright attitude)

Let θ_t be in radians and upright corresponds to $\theta^* = \frac{\pi}{2}$. The tilt error is $\Delta\theta_t = \theta_t - \theta^*$. We penalize absolute tilt:

$$r_\theta(s_t) = -c_\theta\,|\Delta\theta_t|.$$

Leg contact bonus

Binary contact indicators $\ell(t), r(t) \in \{0, 1\}$ give a positive reward for legs touching ground:

$$r_{\ell r}(s_t) = c_\ell\,\ell(t) + c_r\,r(t).$$

Default: $c_\ell = c_r = 10$ (each contact yields +10 as per specification).

Control (fuel) penalty

Let $u_{\text{side}}(t)$ be an indicator (1 if a side thruster fires this frame, else 0) and $u_{\text{main}}(t)$ likewise for the main engine. Apply small per-frame control costs:

$$r_{\text{ctrl}}(a_t) = -c_s\,u_{\text{side}}(t) - c_m\,u_{\text{main}}(t).$$

Per specification choose $c_s = 0.03$ and $c_m = 0.3$ (units: reward points per frame).

Terminal bonus / penalty

At termination (when the simulation ends) apply:

$$r_{\text{term}} = \begin{cases} +100, & \text{if a safe landing is detected (both legs, on-pad, upright),} \\ -100, & \text{otherwise (crash).} \end{cases}$$

Combined explicit formula

Collecting terms, an explicit per-step reward is:

$$\boxed{\begin{aligned} r_t = & - c_{\Delta x}\,\Delta d_t - c_{\Delta v}\,\Delta v_t \\ & - c_\theta \left|\theta(t) - \tfrac{\pi}{2}\right| + c_\ell\,\ell(t) + c_r\,r(t) \\ & - c_s\,u_{\text{side}}(t) - c_m\,u_{\text{main}}(t). \end{aligned}}$$

Terminal reward $r_{\text{term}} \in \{+100, -100\}$ is added once at episode end as described above.

Recommended default coefficients (tunable)

$$\begin{aligned} &c_x = 0.3, \qquad c_v = 0.5, \qquad c_\theta = 2.0, \\ &c_\ell = c_r = 10.0, \qquad c_s = 0.03, \qquad c_m = 0.3. \end{aligned}$$

These values reflect the verbal specification and are a reasonable starting point. Tune them empirically: increase c_v to punish fast landings, or increase c_θ to force a stricter upright policy.

Alternative functional forms (optional)

- Use squared terms for distance/angle (e.g., $-c_x(x - x_{\text{pad}})^2$ or $-c_\theta(\Delta\theta)^2$) to penalize large deviations more strongly.
- Separate horizontal and vertical velocity penalties to prioritize reducing vertical speed at touchdown.