MATH 514 Report: Lunar Lander Optimal Control

Sequoyah Walters

1 Introduction

This project is inspired by the Lunar Lander Atari game released in the late 1970s, in which the angle and thrust of a space craft is controlled in order to land at a precise location. The game is over when the lander crashes into the ground, or when it runs out of fuel. The setup of this game is perfect for solving an optimal control problem – what is the optimal trajectory the lander can take to safely land, and also minimize fuel usage?

Figure 1: Screenshot of the original Atari Lunar Lander arcade game. (source: Wikipedia)

Solving this optimal control problem is not trivial, but there are multiple ways to modify/approximate the problem such that it is tractible. Specifically, [1] shows a direct collocation method that can be utilized to approximate the integrals that arise in an optimal control nonlinear program (NLP). Direct collocation discretizes the problem into separate collocation points, allowing for usage of various approximation techniques in order to approximate the original NLP, such as trapezoidal quadrature or Gaussian quadrature.

After determining an optimal trajectory from solving the NLP, we can then consider open-loop or closed-loop control. Open-loop control can be subject to slight uncertainties in the dynamic model or noise in the state estimation. We show that by using closed-loop control the trajectory is able to closely follow the optimal trajectory even if the open-loop control starts to diverge.

In this report, the outline is as follows: Section 2 considers the dynamic model of the lunar lander. Section 3 formulates the problem statement and introduces the NLP formulation. Section 4 introduces the numerical methods used in this project, such as direct collocation and Runge-Kutta 4. Section 5 shows simulation results, and finally Section 6 concludes with a discussion.

2 Model: Lunar Lander

The model of the lander is now considered: first without drag, then with drag.

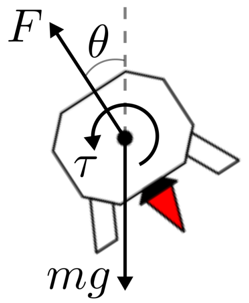

Figure 2: Force diagram of the lunar lander model. Here, we do not yet consider drag forces. F is the thrust force, θ is the angle from the vertical, τ is the torque and mg is mass times gravity (weight). In this model F and τ are the control inputs. All values are in S.I. units.

2.1 Without Drag

Consider a two-dimensional lunar lander system in which a thrust (F) and torque (τ) can be applied as control inputs. This system has the following equations of motion if we initially neglect drag forces:

$$\begin{aligned} m\dot{v}_x &= -F\sin(\theta) \\ m\dot{v}_y &= F\cos(\theta) \\ I\dot{\omega} &= \tau, \end{aligned}$$

where the mass m and the rotational intertia I are constants. The control inputs to the system are thrust F and torque τ, while the system states are horizontal position x, vertical position y, angle θ, horizontal velocity v_x, vertical velocity v_y and angluar velocity ω. Note that a dot above a state variable (e.g. $\dot{v}_x$) denotes the first time derivative of that variable (e.g. horizontal acceleration). Note that Figure 2 is a force diagram for the no-drag case. We consider drag in the following.

2.2 With Drag

The assumption that drag is present in the system is used for the remainder of this report. Here, drag forces are assumed to be linear in velocity. The drag coefficient corresponding to horizontal and vertical motion is b_v, while the rotational drag coefficient is b_ω. For example, the drag force on the x-axis is $-b_v v_x$. As such, we can arrive at the following state dynamics:

$$\begin{aligned} \dot{x} &= v_x \qquad \dot{y} = v_y \qquad \dot{\theta} = \omega \\ \dot{v}_x &= -\frac{1}{m}(F\sin(\theta) + b_v v_x) \\ \dot{v}_y &= \frac{1}{m}(F\cos(\theta) - b_v v_y) - g \\ \dot{\omega} &= \frac{1}{I}(\tau - b_\omega \omega) \end{aligned}$$

Now we can write the state-space description of the dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-1}{m} b_v & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{m} b_v & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{I} b_\omega \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{-1}{m} \sin(\theta) & 0 \\ \frac{1}{m} \cos(\theta) & 0 \\ 0 & \frac{1}{I} \end{bmatrix} \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{1}$$

This can be written as a general control-affine system:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + g(\boldsymbol{x}(t))u(t), \tag{2}$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v_x & v_y & \omega \end{bmatrix}^T$, and $u = \begin{bmatrix} F & \tau \end{bmatrix}^T$. Note the time argument t was been dropped for ease of notation.

3 Problem Statement

Given the control-affine system dynamics (2), we aim to solve a nonlinear program (NLP) optimal control problem by using direct collocation methods with trapezoidal quadrature. The objective of the control problem will be to minimize the total "fuel" usage (i.e. minimize control) of the lander while achieving some reference state $\boldsymbol{x}_{ref}$ with an initial state of $\boldsymbol{x}_{init}$. Additionally the control values $u(t)$ are lower and upper bounded by u_{min} and u_{max}, respectively. The original NLP optimal control problem is as follows:

$$\min_{\boldsymbol{x}(t),u(t)} \quad \int_{t_0}^{t_N} u(t)^T u(t) dt \tag{3a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + g(\boldsymbol{x}(t))u(t) \quad \forall t \in [t_0, t_N] \tag{3b}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_{init} \tag{3c}$$

$$\boldsymbol{x}(t_N) = \boldsymbol{x}_{ref} \tag{3d}$$

$$y(t) \geq d(\theta(t)) \quad \forall t \in [t_0, t_N] \tag{3e}$$

$$u_{min} \leq u(t) \leq u_{max} \quad \forall t \in [t_0, t_N], \tag{3f}$$

where $d(\theta)$ is specified in Figure 3.

3.1 Explanation of NLP (3)

Walking through each part of (3), first note that (3a) minimizes the control over the whole time period $t \in [t_0, t_N]$, due to the integral. Constraint (3b) restricts the decisions functions $\boldsymbol{x}(t)$ and $u(t)$ to abide by the system dynamics from (2). Constraints (3c) and (3d) define the initial and final boundary conditions for this NLP, respectively. Constraint (3e) ensures that the legs of the lander do not go below the ground. Furthermore, $d(\theta)$ is the distance from the center of the lander to the ground, when at least one leg is touching the ground (see Figure 3). Lastly, constraint (3f) ensures the control inputs to the system remain bounded.

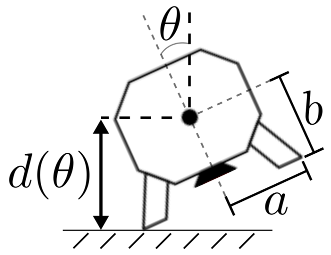

Figure 3: Dimension of lander. This information is used to ensure neither of the legs go below the floor when the optimal trajectory is computed. In this way, we require $y(t) \geq d(\theta(t)) = \sin\left(|\theta(t)| + \sin^{-1}\left(\frac{b}{\sqrt{a^2+b^2}}\right)\right)\sqrt{a^2+b^2}$ for all time $t \in [t_0, t_N]$. Straight-forward geometry was used to determine the function $d(\theta)$.

4 Numerical Methods

To solve the NLP (3) exactly is very difficult due to the continuous-time dynamics which causes the integral in the objective and the non-discrete system dynamics constraint. In order to solve this problem, we consider direct collocation which discretizes the dynamics, allowing for the approximation of the integral in the objective, and other integrals that result from the system dynamics constraint. We will then consider Runge-Kutta 4 for the simulation of the closed loop system, tracking the optimal trajectory.

4.1 Direct Collocation

In direct collocation, we can choose N collocation points (in this case time points). In this way, there are N decision variables for each state, and N for each control input. If n is the number of states, and m is the number of controls, the total number of decision variables is then $N(n+m)$. We first talk about direct collocation using trapezoidal quadrature.

4.1.1 Trapezoidal Quadrature

Using trapezoidal quadrature, if we want to estimate the value of an integral we split the in-egral up so each one is evaluated between two collocation points, then use the trapezoidal rule to approximate each integral. To demonstrate, consider integrating an arbitrary Lipschitz function $q(t) : \mathbb{R} \to \mathbb{R}$ as follows:

$$\int_{t_0}^{t_N} q(T)dT = \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} q(T)dT \approx \sum_{k=0}^{N-1} \frac{1}{2} h_k [q(t_k) + q(t_{k+1})],$$

where $h_k = t_{k+1} - t_k$. In this way, the objective function (3a) then can be approximated as:

$$\begin{aligned}\int_{t_0}^{t_N} u(t)^T u(t)dt &\approx \sum_{k=0}^{N-1} \frac{1}{2} h_k \Big(u(t_k)^T u(t_k) + u(t_{k+1})^T u(t_{k+1}) \Big) \\ &= \sum_{k=0}^{N-1} \frac{1}{2} h_k \Big(F(t_k)^2 + F(t_{k+1})^2 + \tau(t_k)^2 + \tau(t_{k+1})^2 \Big). \end{aligned} \tag{4}$$

Next, the dynamics constraint (3b) can also be approximated using an integral. For ease of notation, $f(\boldsymbol{x}(t))$, $g(\boldsymbol{x}(t))$ are written as $f(t)$ and $g(t)$. When we evaluate a function at a specific time point, the time becomes a subscribt (e.g. $x(t_k) := x_k$) in the following:

$$\begin{aligned}
&\dot{\boldsymbol{x}}(t) = f(t) + g(t)u(t) \quad \forall t \in [t_0, t_N] \\
\implies &\int_{t_k}^{t_{k+1}} \dot{\boldsymbol{x}}(t)dt = \int_{t_k}^{t_{k+1}} (f(t) + g(t)u(t))dt \quad \forall k \in \{0, 1, \dots, N-1\} \\
\implies &\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \approx \frac{1}{2}h_k(f_k + g_k u_k + f_{k+1} + g_{k+1}u_{k+1}) \quad \forall k \in \{0, 1, \dots, N-1\}. \qquad (5)
\end{aligned}$$

Trivially, the boundary conditions (3c) and (3d) remain the same, with $\boldsymbol{x}_0 = \boldsymbol{x}_{init}$ and $\boldsymbol{x}_N = \boldsymbol{x}_{ref}$. Additionally, the final two constraints remain the same, except they must hold for each collocation time point, and not necessarilly all time in the interval. Putting all of these new constraints together, we arrive at the direct collocation optimal control problem using trapezoidal quadrature. We let h_k be constant for all k values, so $h := h_k$.

Trapezoidal Quadrature Collocation NLP:

$$\begin{aligned}
\min_{\boldsymbol{x}_k, u_k} \quad & \sum_{k=0}^{N-1} \left(F_k^2 + F_{k+1}^2 + \tau_k^2 + \tau_{k+1}^2\right) && \text{(6a)} \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \frac{1}{2}h(f_k + g_k u_k + f_{k+1} + g_{k+1}u_{k+1}) \quad \forall k \in \{0, 1, \dots, N-1\} && \text{(6b)} \\
& \boldsymbol{x}_0 = \boldsymbol{x}_{init} && \text{(6c)} \\
& \boldsymbol{x}_N = \boldsymbol{x}_{ref} && \text{(6d)} \\
& y_k \geq d(\theta_k) \quad \forall k \in \{0, 1, \dots, N\} && \text{(6e)} \\
& u_{min} \leq u_k \leq u_{max} \quad \forall k \in \{0, 1, \dots, N\}. && \text{(6f)}
\end{aligned}$$

Note that contstraint (6b) actually contains n constraints (the number of states) for each value of k.

4.1.2 Gaussian Quadrature

Direct collocation using Gaussian quadrature is also possible, and benefits from a higher order accuracy result. Its main benefit is being able to choose the evaluation points, allowing for the cancellation of the derivatives of the function that is being integrated. This allows for nice accuracy, without need derivative information. This is done by choosing the evaluation points as the roots of an orthogonal polynomial. Cancellation ensuses because the inner product of two orthogonal polynomials is zero. Unfortunatley for this project, formulating all the constraints for the nonlinear program was a challenge that could not be overcome at this time. Therefore, trapezoidal quadrature is used in the results section.

4.2 Interpolation

Once we have the (near) optimal trajectory determined by solving a collocation based NLP, we will want to interpolate the state and control values. This will allow the subsequent simulation to use values that are in between the collocation points. Specifically for this project, linear and quadratic interpolation sufficiently allowed to simulation to run smoothly.

Another reason for interpolation is to be able to employ a state-feedback controller to to track the optimal state trajectory. This closed-loop approach can help account for any difference that arises between the open-loop trajectory and the optimal trajectory.

4.3 Runge-Kutta 4

After solving the optimal control problem with a direct collocation method, we now still need to simulate the lander system using the trajectory that we obtained. For this, we use Runge-Kutta 4 (RK4), which is quite easy to implement and has 4-th order accuracy. RK4 was chosen for this project because the lunar lander system is not stiff, and has quite smooth system dynamics. In this way RK4 is more than accurate enough to handle this system. The course text [2] delves deeper into RK4 and other RK algorithms.

The RK4 algorithm works as follows, where $F(\cdot)$ is a vector of differential equations corresponding to the vector of states X, and h is the time step size:

$$\begin{aligned}
k_1 &= F(X(t)) \\
k_2 &= F(X(t) + \frac{1}{2}hk_1) \\
k_3 &= F(X(t) + \frac{1}{2}hk_2) \\
k_4 &= F(X(t) + hk_3) \\
k &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
X(t+h) &= X(t) + hk
\end{aligned}$$

5 Results

The code and some videos of the results are available online at `https://github.com/seqwalt/LunarLander`. In this section, we examine some of the results from the proposed methods. Specifically, we will use trapezoidal quadrature with direct collocation to form a NLP. This NLP is then solved with IPOPT using the python interface pyomo. Then, the optimal trajectory is interpolated, and a closed-loop feedback controller is used in simulation in order keep the simulated system close to the optimal trajectory. This feed-back controller is basically a proportional-derivate linear feedback controller, and was hand-tuned.

5.1 Open-Loop vs. Closed-Loop Control

To note the discrepancy between open and closed loop control, Figure 4 shows an open-loop trajectory simulation, in which the simulated system starts to deviate from the optimal trajectory. However after applying the closed-loop controller, we can see in Figure 5 that there is a noticeable improvement in the tracking of the optimal trajectory.

5.2 Low vs. High Gravity

Changing the level of gravity produced interesting results. Given a fixed time constraint to achieve the goal state, under low gravity the lander would make high-angle turns as seen in Figure 6, where gravity was set to $0.3\frac{m}{s^2}$. However under higher gravity, this lander would stay largely near an angle of $\theta = 0$. In Figure 7, the lander stays close to the ground, and the gravity is set to $9.8\frac{m}{s^2}$.

5.3 Doing a Flip

In Figure 8 the lander performs one full rotation and is able to land. This is accomplished by setting the initial and to $\theta_0 = 2\pi$ and the reference angle to $\theta_N = 0$, forcing the NLP to flip the lander. The lander has been tested to be able to do up to 5 flips before the NLP has trouble finding a feasible solution. This might be aided by implementing a more advanced quadrature method in the future, such as guassian quadrature.

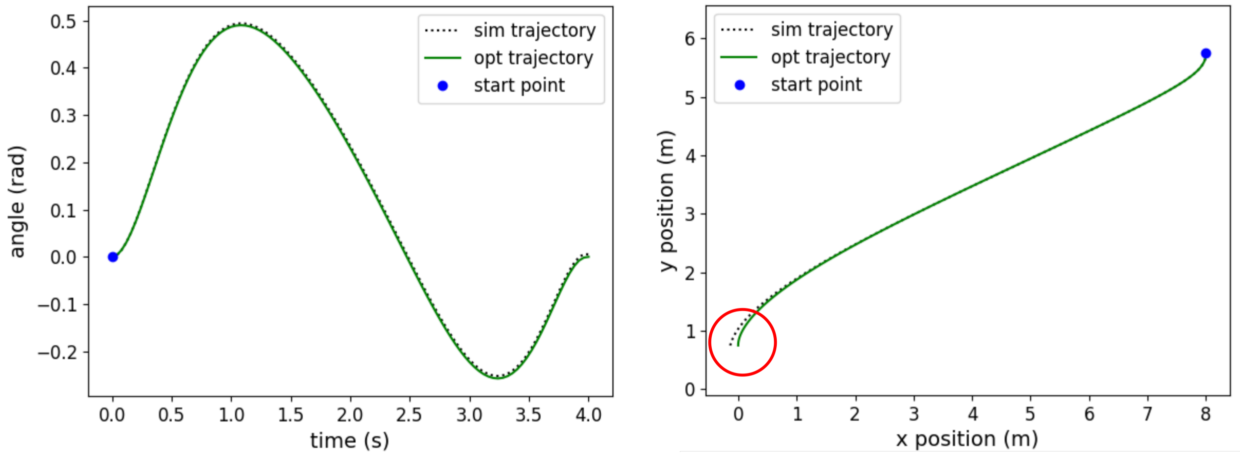

Figure 4: Open loop trajectory of the lunar lander system. Note how at the tail end of the trajectory, the simulation (black dots) starts to stray from the optimal trajectory (green line). This is highlighted within the red circle. The left plot is angle over time, and the right plot is position.

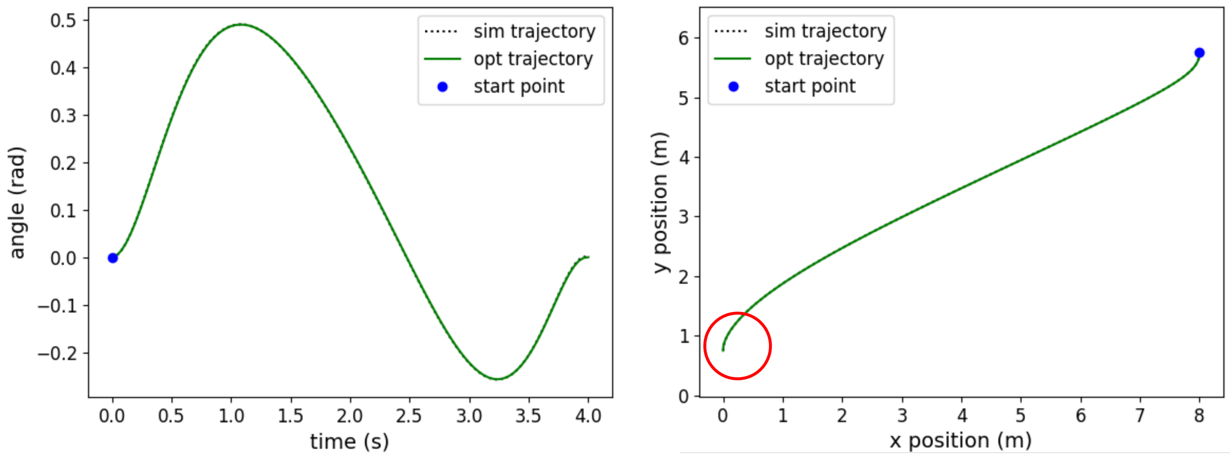

Figure 5: Closed loop trajectory of the lunar lander system. Note how through the whole trajectory, the simulation (black dots) stays directly on top of the optimal trajectory (green line) – even at the end of the trajectory. This is highlighted within the red circle. The left plot is angle over time, and the right plot is position.

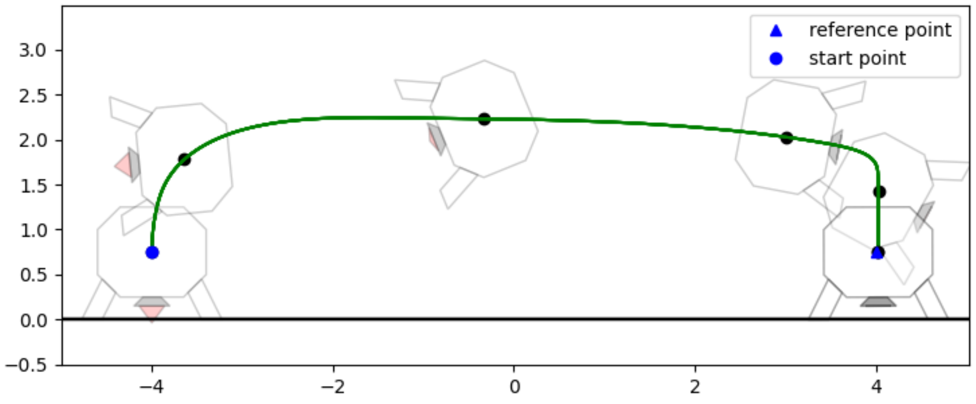

Figure 6: Low gravity example, going from left to right. The gravity is set to $g = 0.3\frac{m}{s^2}$.

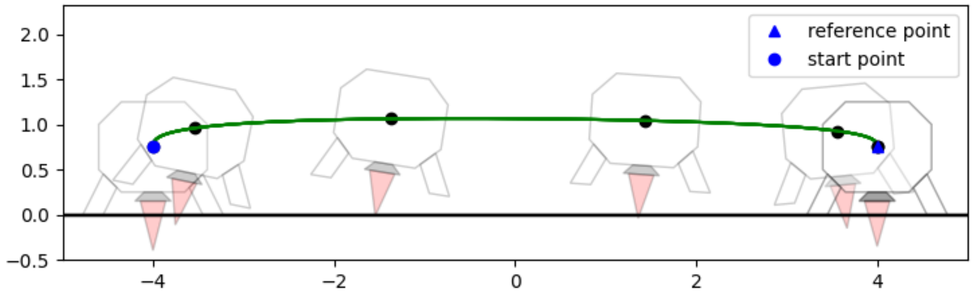

Figure 7: High gravity example, going from left to right. The gravity is set to $g = 9.8\frac{m}{s^2}$.

6 Conclusion

In this report we have presented a direct collocation method that can be used in conjucture with a nonlinear program. The direct collocation method disrectizes the dynamic allowing to estimate the integrals that arise in the NLP. An interpolation and then simulation is done with a closed-loop state-feedback controller, which tracks the optimal trajectory that was generated by the NLP. Future directions could include implementing Gauss quadrature instead of trapezoidal quadrature. Additionally, it would be interesting to apply this method to a multi-agent system in which obstacle avoidance is required to not collide with other agents.

References

[1] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[2] Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

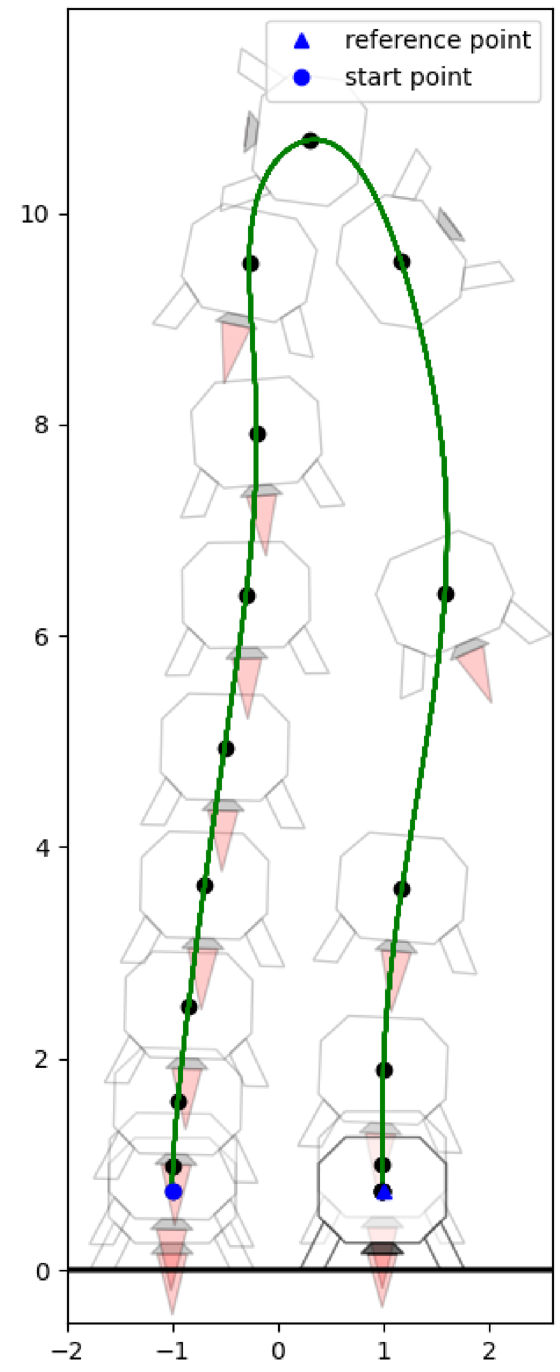

Figure 8: A full rotation of the lander is achieved by setting $\theta_0 = 2\pi$ and $\theta_N = 0$. This causes the NLP to flip the lander. In this example, the lander starts on the left, and ends on the right.