Three-Stage Approach to Optimal Low-Thrust Earth-Moon Trajectories With Variable Thrust Magnitude

by

Tushar Sial & Aman Singh

A Project report submitted to the graduate faculty

in partial fulfillment of the requirements for the course

AERE 6510 - Spacecraft Trajectory Optimization

Major: Aerospace Engineering

Professor:
Dr. Ossama Abdelkhalik

Iowa State University

Ames, Iowa

2025

TABLE OF CONTENTS

ACKNOWLEDGMENTS

We want to thank those whose help has been instrumental in the culmination of this Project Report. Firstly, to Dr. Ossama Abdelkhalik for his inspirational teaching style, guidance, and encouragement that have equipped us with the advanced concepts in Spacecraft Trajectory Optimization, required to complete this project. We would also like to thank the Teaching Assistant, Aimar Negrete, for his patience and support throughout the project and the course duration. Lastly, we extend our gratitude to the pioneers of orbital mechanics, whose foundational research propelled the innovation in spaceflight and orbital mechanics.

ABSTRACT

This project explores and extends the three-stage approach for computing optimal low-thrust trajectories from low Earth orbit (LEO) to low lunar orbit (LLO) as initially proposed by Pierson and Kluever. The original three-stage method systematically decomposes the complex optimal trajectory problem into simpler subproblems involving maximum-energy Earth-escape and moon-capture spirals, an all-coasting translunar trajectory, and finally, a complete optimization using a hybrid direct/indirect numerical method. In our study, we revisit and recreate the original three-stage methodology while extending its scope by allowing the thrust magnitude to vary freely, rather than remain fixed, thereby potentially enhancing fuel efficiency and flexibility of mission profiles.

The classical restricted three-body problem dynamics govern the spacecraft trajectory, involving highly sensitive numerical computations and optimization strategies. Our extended methodology employs advanced optimization techniques to explore a broader solution space, providing comparative analyses between fixed and variable thrust magnitude scenarios. The numerical results indicate significant implications for mission design flexibility, trajectory efficiency, and overall spacecraft mass delivered to lunar orbit. This enhanced framework offers valuable insights for future low-thrust lunar missions, potentially reducing costs and increasing payload capacity by effectively optimizing trajectory and thrust parameters simultaneously.

CHAPTER 1. General Introduction

1.0.1 Introduction

The exploration of space has always been driven by the quest for efficiency and effectiveness in mission design, particularly in terms of fuel consumption, payload capacity, and overall mission cost. Among various propulsion techniques available, low-thrust propulsion systems stand out due to their ability to deliver higher payload fractions compared to conventional chemical propulsion, despite their inherently longer mission durations.

Low-thrust propulsion systems, such as electric propulsion, offer continuous thrust capability, enabling spacecraft to perform gradual orbital maneuvers that are particularly advantageous for interplanetary missions. The feasibility and efficiency of low-thrust trajectories are well-established, making them a preferred choice for long-duration missions to the Moon, Mars, and beyond.

This study builds upon the foundational work of Pierson and Kluever, who proposed a systematic three-stage approach to solving optimal low-thrust trajectories from low Earth orbit (LEO) to low lunar orbit (LLO). Their methodology, rooted in the classical restricted three-body problem, addresses the complexities and sensitivities inherent in trajectory optimization by breaking down the problem into manageable stages. This involves initial maximum-energy trajectory segments, followed by an efficient all-coasting translunar phase, and culminating in a comprehensive optimization stage using a hybrid direct/indirect approach.

Our research extends this proven framework by considering scenarios where the thrust magnitude is not constrained, thereby introducing a new dimension of flexibility into trajectory optimization. By exploring variable thrust magnitudes, this project aims to enhance the adaptability and efficiency of lunar missions, potentially reducing fuel requirements and allowing greater payload capacities.

In the subsequent chapters, the study's methodologies, computational models, numerical simulations, and comparative analyses between fixed and variable thrust scenarios will be elaborated. This comprehensive examination seeks not only to validate the original three-stage approach but also to demonstrate significant advancements possible through the introduction of variable thrust magnitude in lunar mission design.

1.0.2 Methodology

We decompose the full LEO→LLO minimum-fuel problem into three subproblems:

1. **Stage 1:** Max-energy Escape & Capture spirals
2. **Stage 2:** Suboptimal Translunar Coast
3. **Stage 3:** Hybrid Direct/Indirect Optimization

In all powered arcs, the thrust magnitude T is now treated as an *optimizable* variable within engine limits $0 \le T \le T_{\max}$, rather than fixed as in the original paper.

1.0.2.1 Stage 1: Maximum-Energy Escape & Capture

Goal: For a given burn duration t_f, find

$$u(t),\ T \quad \Longrightarrow \quad \max\Big[\tfrac{1}{2}(v_r^2 + v_\theta^2) - \tfrac{\mu}{r}\Big]_{t_f} \ \equiv\ \min J \ =\ -E(t_f).$$

Design variables:
- Thrust-direction, $u(t)$, $0 \le t \le t_f$
- Thrust magnitude, $T \in [0, T_{\max}]$

Boundaries:
- *Escape:* Start at $r = r_{\text{LEO}}$, $v_r = 0$, $v_\theta = \sqrt{\mu/r_{\text{LEO}}}$.
- *Capture:* Back-integrate from $r = r_{\text{LLO}}$, circular velocity at LLO.

Solution: Indirect (costate) BVP + SQP, sweeping t_f to build lookup tables of (v_r, v_θ) vs. r.

1.0.2.2 Stage 2: Suboptimal Translunar Coast

Goal: Connect end of Stage 1 escape to start of Stage 1 capture with an all-coast transfer that minimizes total powered-time (hence propellant).

Design variables:
- Coast initial state: $r(0)$, $\theta(0)$
- Coast duration t_{coast}
- Final mass $m(t_{\text{coast}})$

Objective:

$$J = t_{\text{esc}}(r(0), m(0)) \; + \; t_{\text{cap}}(r(t_{\text{coast}}), m(t_{\text{coast}})),$$

where t_{esc}, t_{cap} are obtained from Stage 1 curve-fits.

Dynamics: Restricted three-body (Earth+Moon), *no thrust*.

Constraints: Match (v_r, v_θ) at start/end of coast to Stage 1 values.

Solution: Direct SQP on 4 variables, 3 equality constraints.

1.0.2.3 Stage 3: Hybrid Direct/Indirect Minimum-Fuel Optimization

Goal: Solve the full LEO→LLO transfer (spiral–coast–spiral) in one NLP, using Stage 2 outputs as an initial guess.

Controls & Unknowns:
- Piecewise-constant thrust magnitude T_i and steering angles $u_i(t)$ on each powered arc ($i = \text{esc}, \text{cap}$).
- Durations t_{esc}, t_{coast}, t_{cap}.
- Initial costates $\lambda(0)$ and final costates $\lambda(t_f)$.

Dynamics: Full restricted three-body with thrust in rotating frames, continuity at sphere-of-influence.

Initial Guess Construction: 1. Pick $t_{\text{esc}}, t_{\text{cap}}$ from Stage 2 coast end.

2. Import costate/time histories from Stage 1 max-energy runs.

Method: Hybrid direct/indirect (costate-based parameterization inside SQP) for robustness and accuracy.

1.0.2.4 Key Note on Thrust

Unlike Pierson and Kluever's fixed-thrust assumption, here each powered arc allows

$$0 \le T \le T_{\max}$$

as a free design variable, enabling a richer trade-off between propellant use and transfer time.

CHAPTER 2. Stage 1: Maximum-Energy Escape/Moon Capture Trajectories

2.0.1 Fixed–End–Time Optimal Earth–Escape Spiral with Free Thrust Magnitude

To examine the behavior of long-duration, continuous-thrust spirals around both Earth and the Moon, we first solve two separate, fixed-duration optimization problems—one for escaping Earth's gravity and one for capturing into lunar orbit. In each case, we seek the thrust profile that delivers the greatest mechanical energy (kinetic + potential) at the end of an outward spiral from Earth or at the start of an inward spiral toward the Moon. By sweeping over a range of allotted times, these "maximum-energy" spirals provide the velocity vectors needed as boundary conditions at the ends of a simpler, all-coast transfer. Although such energy-maximizing segments may not lie on the true minimum-propellant path, they form a readily solvable first step that guides the full minimum-fuel Earth–Moon trajectory.

In contrast to the original formulation—where the thrust magnitude T was fixed—here we introduce T as an additional design variable (subject to any engine limits). The fixed–end–time Earth-escape problem is then stated as follows:

Design Variables:

$u(t)$ Thrust–direction angle, defined for $0 \leq t \leq t_f$.

T Constant thrust magnitude (to be optimized).

Objective:

Maximize the spacecraft's total mechanical energy at the final time t_f, i.e. minimize

$$J \;=\; -\Big[\tfrac{1}{2}\big(v_r^2 + v_\theta^2\big) \;-\; \frac{\mu_E}{r}\Big]_{t=t_f}.$$

Equations of Motion:

In polar coordinates (r, θ) about the Earth:

$$\dot{r} = v_r, \tag{2.1}$$

$$\dot{v}_r = \frac{v_\theta^2}{r} - \frac{\mu_E}{r^2} + \frac{T}{m}\sin\alpha, \tag{2.2}$$

$$\dot{v}_\theta = -\frac{v_r\, v_\theta}{r} + \frac{T}{m}\cos\alpha, \tag{2.3}$$

$$\dot{\theta} = \frac{v_\theta}{r}, \tag{2.4}$$

$$\dot{m} = -\frac{T}{I_{\text{sp}}\, g_0}. \tag{2.5}$$

Initial Conditions:

Assuming a circular low-Earth orbit at $t = 0$:

$$\begin{aligned} r(0) &= r_{\text{LEO}}, \qquad v_r(0) = 0, \\ v_\theta(0) &= \sqrt{\frac{\mu_E}{r_{\text{LEO}}}}, \quad \theta(0) = 0, \\ m(0) &= m_0. \end{aligned}$$

Engine Constraints:

Engine Type: Constant Specific Impulse (CSI)

$$\phi_1 \;=\; T\big(T_{\max} - T\big) \;-\; \gamma^2, \tag{2.6}$$

$$K \;=\; \varepsilon\, J \;+\; \int_{t_0}^{t_1} u\, \phi_1 \,\mathrm{d}t, \tag{2.7}$$

Control, States and Costates:

$$u = \begin{pmatrix} T \\ \alpha \\ \gamma \end{pmatrix}, \quad \tilde{x} = \begin{pmatrix} r \\ V_r \\ V_\theta \\ \theta \\ m \end{pmatrix}, \quad \tilde{\lambda} = \begin{pmatrix} \lambda_r \\ \lambda_{v_r} \\ \lambda_{v_\theta} \\ \lambda_\theta \\ \lambda_m \end{pmatrix}. \tag{2.8}$$

Boundary Conditions at t_f:

The usual two-point boundary-value constraints apply: the costates must satisfy the transversality conditions for free final angle $\theta(t_f)$ and tangent steering at $r(t_f)$.

Once these maximum-energy spirals are computed for both Earth-escape and lunar-capture, their end-states serve as velocity-vector boundary conditions for the subsequent coasting transfer subproblem, which in turn feeds into the full minimum-fuel Earth–Moon solution.

2.0.2 Necessary Conditions of Optimality

Throughout, the control vector is

$$u = \big(T,\ \alpha,\ \gamma\big), \quad 0 \le T \le T_{\max}\,,$$

so that T enters as an *optimizable* variable.

1. Costate (Adjoint) Equations

$$\dot{\lambda}_r = -\frac{\partial H}{\partial r} = \lambda_{v_r}\left(\frac{V_\theta^2}{r^2} - \frac{2\mu}{r^3}\right) - \lambda_{v_\theta}\frac{V_r V_\theta}{r^2} + \lambda_\theta \frac{V_\theta}{r^2}, \tag{2.9}$$

$$\dot{\lambda}_{v_r} = -\frac{\partial H}{\partial V_r} = -\lambda_r + \lambda_{v_\theta}\frac{V_\theta}{r}, \tag{2.10}$$

$$\dot{\lambda}_{v_\theta} = -\frac{\partial H}{\partial V_\theta} = -2\lambda_{v_r}\frac{V_\theta}{r} + \lambda_{v_\theta}\frac{V_r}{r} - \frac{\lambda_\theta}{r}, \tag{2.11}$$

$$\dot{\lambda}_\theta = -\frac{\partial H}{\partial \theta} = 0, \tag{2.12}$$

$$\dot{\lambda}_m = -\frac{\partial H}{\partial m} = \lambda_{v_r}\frac{T\sin\alpha}{m^2} + \lambda_{v_\theta}\frac{T\cos\alpha}{m^2}. \tag{2.13}$$

2. Transversality at $t = t_1$

Define the Mayer term $G\big(t_1, \tilde{x}(t_1)\big) = -\varepsilon\big[\frac{1}{2}(V_r^2 + V_\theta^2) - \frac{\mu}{r}\big]_{t_1}$. Then

$$\lambda_r(t_1) = \frac{\partial G}{\partial r}\bigg|_{t_1} = -\frac{\mu}{r_1^2}, \tag{2.14}$$

$$\lambda_{v_r}(t_1) = \frac{\partial G}{\partial V_r}\bigg|_{t_1} = -\varepsilon\, V_r(t_1), \tag{2.15}$$

$$\lambda_{v_\theta}(t_1) = \frac{\partial G}{\partial V_\theta}\bigg|_{t_1} = -\varepsilon\, V_\theta(t_1), \tag{2.16}$$

$$\lambda_\theta(t_1) = \frac{\partial G}{\partial \theta}\bigg|_{t_1} = 0, \tag{2.17}$$

$$\lambda_m(t_1) = \frac{\partial G}{\partial m}\bigg|_{t_1} = 0. \tag{2.18}$$

3. Stationarity of the Hamiltonian

Let $c = I_{\mathrm{sp}}\, g_0$ and ν be the multiplier for the constraint $T(T_{\max} - T) - \gamma^2 = 0$. Then

$$\frac{\partial H}{\partial \alpha} = 0 \quad \Longrightarrow \quad \lambda_{v_r}\frac{T}{m}\cos\alpha - \lambda_{v_\theta}\frac{T}{m}\sin\alpha = 0, \tag{2.19}$$

$$\frac{\partial H}{\partial T} = 0 \quad \Longrightarrow \quad \lambda_{v_r}\frac{\sin\alpha}{m} + \lambda_{v_\theta}\frac{\cos\alpha}{m} - \frac{\lambda_m}{c} + \nu\big(T_{\max} - 2T\big) = 0, \tag{2.20}$$

$$\frac{\partial H}{\partial \gamma} = 0 \quad \Longrightarrow \quad 2\,\gamma\,\nu = 0. \tag{2.21}$$

Solving $\partial H/\partial\alpha = 0$ for α gives

$$T = 0 \text{ or } \tan\alpha \;=\; \frac{-\lambda_{v_r}}{-\lambda_{v_\theta}} \quad\Longrightarrow\quad T = 0 \text{ or } \alpha^*(t) = \arctan\!\Big(\tfrac{-\lambda_{v_r}(t)}{-\lambda_{v_\theta}(t)}\Big).$$

Next, we compute the Hessian of the Hamiltonian wrt the control vector to satisfy the Legendre-Clebsch condition $\partial^2 H/\partial u^2 \succeq 0$:

$$\partial^2 H/\partial u^2 = \begin{bmatrix} -\lambda_{v_r}\frac{T}{m}\sin\alpha - \lambda_{v_\theta}\frac{T}{m}\cos\alpha & \lambda_{v_r}\cos\alpha - \lambda_{v_\theta}\sin\alpha & 0 \\ \lambda_{v_r}\cos\alpha - \lambda_{v_\theta}\sin\alpha & -2\nu & 0 \\ 0 & 0 & -2\nu \end{bmatrix} \tag{2.22}$$

By analysing eqn 2.21, we get the two cases:

Case 1: $\gamma = 0,\ \nu \neq 0$ **(bang–bang).** In the general (non-singular) case, the optimal thrust is bang–bang:

$$T(t) \;=\; \begin{cases} T_{\max}, & \chi(t) > 0, \\ 0, & \chi(t) < 0. \end{cases}$$

For this case, the Hessian simply becomes:

$$\partial^2 H/\partial u^2 = \begin{bmatrix} -\lambda_{v_r}\frac{T}{m}\sin\alpha^* - \lambda_{v_\theta}\frac{T}{m}\cos\alpha^* & 0 & 0 \\ 0 & -2\nu & 0 \\ 0 & 0 & -2\nu \end{bmatrix} \tag{2.23}$$

In order for this to be positive semidefinite, all the eigenvalues have to be non-negative, which gives us the negative sign for the steering law as well as a condition on ν:

$$\alpha^*(t) = \arctan\!\Big(\tfrac{-\lambda_{v_r}(t)}{-\lambda_{v_\theta}(t)}\Big) \quad \& \quad \nu^* < 0 \tag{2.24}$$

Case 2: $\gamma \neq 0,\ \nu = 0$ **(singular arc).** On a singular arc one has $\chi(t) = 0$ and $0 < T < T_{\max}$. Solving $\chi = 0$ for T yields

$$T \;=\; \frac{c^2}{\lambda_m\,(c-1)}\Big[\big(-\lambda_{v_r} + \tfrac{\lambda_{v_\theta}V_\theta}{r}\big)\sin\alpha \;+\; \big(-2\,\lambda_{v_\theta}\tfrac{V_\theta}{r} + \tfrac{\lambda_{v_r}V_\theta}{r}\big)\cos\alpha\Big],$$

where $c = I_{\text{sp}}\,g_0$.

Thrust Control Law with Free Thrust Magnitude:

With T now treated as a free control variable ($0 \leq T \leq T_{\max}$), the switching function for thrust is derived by applying Pontryagin's Maximum principle:

$$\chi(t) \; = \lambda_{v_r} \frac{\sin \alpha}{m} + \lambda_{v_\theta} \frac{\cos \alpha}{m} - \frac{\lambda_m}{c}$$

This switching function is used to determine which of the 2 cases is active at any particular time during the powered flight.

2.0.3 Solution of the Two-Point Boundary-Value Problem and Auxiliary Spiral Generation

Once the stationarity condition is applied, the full two-point boundary-value problem (2PBVP) consists of:

- The state equations (Eqs. 2.1-2.5) and costate dynamics (Eqs. 2.9-2.13).
- Transversality constraints at t_f (Eqs. 2.14–2.18).
- The thrust magnitude T promoted to an optimizable constant subject to

$$0 \; \leq \; T \; \leq \; T_{\max}.$$

The unknowns are the initial adjoint variables and the thrust level T.

Numerical Solver: We employ Sequential Quadratic Programming (SQP), which enforces the terminal conditions by minimizing a terminal-error index. Gradients are computed via first-order finite differences, so no analytic derivatives are required.

Earth-Escape Spirals: For each prescribed burn duration t_f, SQP adjusts the costates and the free thrust T until both the free–final–angle and tangent–steering transversality conditions are satisfied. Sweeping t_f yields a family of maximum-energy escape spirals, each characterized by terminal radius $r(t_f)$, velocity components (v_r, v_θ), and escape time t_{esc}.

Moon-Capture Spirals: By integrating backward in time from a circular low-lunar orbit—treating the LLO mass m_{LLO} as a parameter and allowing T to vary—we similarly obtain maximum-energy capture spirals over a grid of $\{r_{\text{LLO}}, m_{\text{LLO}}\}$ values.

Curve Fitting: The datasets

$$\{\ v_r(t_{esc}),\ v_\theta(t_{esc}),\ t_{\text{esc}}\},\quad \{\ v_r,\ v_\theta,\ t_{\text{cap}}\},$$

parameterized by final radius for the capture data set and by final radius and mass at LLO, are smoothed via variable polynomial and cubic spline fits respectively. These mappings

$$(r, m)\ \mapsto\ (v_r, v_\theta, t)$$

then serve as ultra-fast boundary-condition look-ups for the Stage 2 coasting optimization.

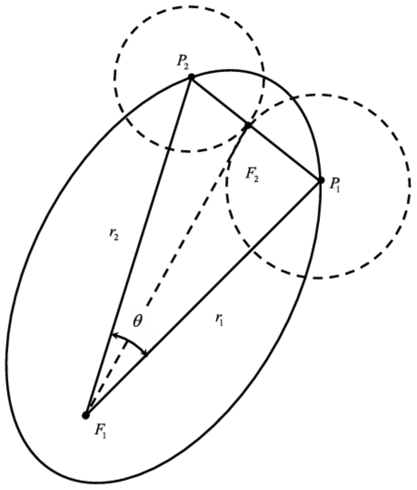

Figure 2.1: Maximum-energy Earth-escape spiral for $t_f = 2.37$ days.

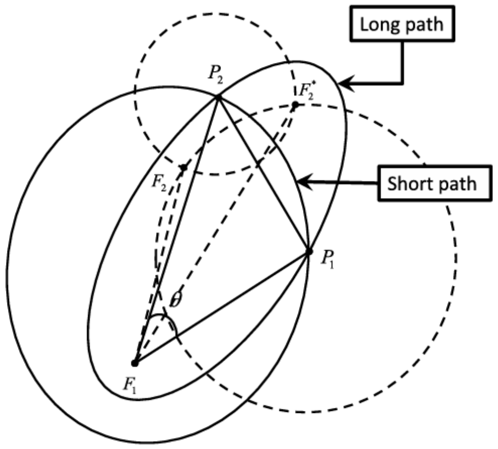

Figure 2.2: Optimal thrust direction and flight path angles: Earth-Escape Phase.

CHAPTER 3. Stage 2: Suboptimal Coasting Translunar Trajectories

Having generated families of maximum-energy spiral arcs in Stage 1-each with its own optimal thrust magnitude $T \in [0, T_{\max}]$, we now splice those powered segments with an unpowered "coast" arc. The goal of Stage 2 is to choose the coast parameters so as to minimize the sum of the two spiral burn times, using Stage 1 curve-fits to supply the exact boundary-condition data.

1. Decision Variables and Objective:

$$x = \big[r_0,\ \theta_0,\ t_c,\ m_f\big], \qquad J(x) = t_{\text{esc}}\big(r_0,\ m_0\big)\ +\ t_{\text{cap}}\big(r_f,\ m_f\big).$$

Here:

- r_0, θ_0: radius and true anomaly at coast start.
- t_c: coast duration.
- m_f: spacecraft mass at coast end.
- $m_0 = m_{\text{LEO}} - \Delta m_{\text{esc}}(r_0)$, and $r_f = r(t_c)$, $\theta_f = \theta(t_c)$.
- $t_{\text{esc}}(r_0, m_0)$ and $t_{\text{cap}}(r_f, m_f)$ are obtained from polynomial/spline fits to the Stage 1 data—which were themselves generated under a free-T thrust optimization.

One thing to note is, that even though our main objective is to minimize fuel consumption, for this subproblem, we aimed at minimizing the time of powered flight and the reason behind this was that from the plots of the first subproblem, we realised that maximum thrust was being applied at the escape and capture spiral phase for 99% of the duration.

2. Dynamics (Unpowered): We integrate the planar restricted three-body polar equations in the Earth-centered rotating frame:

$$\begin{aligned}
&\dot r = v_r, && \dot\theta = \frac{v_\theta}{r},\\
&\dot v_r = \frac{v_\theta^2}{r} - 2\omega\, v_\theta - \mu_E\,\frac{r - D\cos\theta}{\big[r^2 + D^2 - 2Dr\cos\theta\big]^{3/2}} - \mu_M\,\frac{r - D\cos\theta}{\big[r^2 + D^2 - 2Dr\cos\theta\big]^{3/2}}, &&\\
&\dot v_\theta = -\frac{v_r\, v_\theta}{r} + 2\omega\, v_r, && \dot m = 0,
\end{aligned}$$

with ω = Earth–Moon synodic rate, D = Earth–Moon separation.

3. Boundary-Condition Maps: From Stage 1 we have smooth mappings

$$\begin{gathered}
h_1 : (r_0, m_0) \mapsto \theta_0, \quad h_2 : (r_0, m_0) \mapsto \Delta m_{\text{esc}},\\
g_1 : (r_f, m_f) \mapsto v_r(t_c), \quad g_2 : (r_f, m_f) \mapsto v_\theta(t_c),\\
\tau_{\text{esc}} : (r_0, m_0) \mapsto t_{\text{esc}}, \quad \tau_{\text{cap}} : (r_f, m_f) \mapsto t_{\text{cap}}.
\end{gathered}$$

These were obtained by fitting polynomials (or splines) to the Stage 1 outputs $\{r, v_r, v_\theta, t\}$, each computed under its own optimal thrust T.

4. Constraints:

Initial mass:	$m_0 = m_{\text{LEO}} - h_2(r_0, m_0)$,
Initial anomaly:	$\theta(0) = h_1(r_0, m_0)$,
Terminal velocity matching:	$v_r(t_c) = g_1\big(r_f, m_f\big), \quad v_\theta(t_c) = g_2\big(r_f, m_f\big)$,
Mass consistency:	$m(t_c) = m_f$.

5. Numerical Solution via Direct SQP:

- We solve $\min_x J(x)$ subject to the four nonlinear equalities above.
- The variables $r_0 \in [r_{\text{LEO}}, r_{\text{SOI}}]$, $\theta_0 \in [0, 2\pi)$, $t_c > 0$, $m_f \in [m_{\text{LEO}} - \Delta m_{\max}, m_{\text{LEO}}]$ are bounded.
- We employ a direct SQP solver (MATLAB's `fmincon`) with finite-difference gradients.

- Typical convergence is achieved in 5–10 SQP iterations, even from crude initial guesses.

6. Discussion of Free-T Influence: By allowing T to vary in Stage 1, the boundary-condition maps h_i, g_i, and $\tau_{(\cdot)}$ capture the trade-off between thrust-level and burn time. As a result:

- The coast solver automatically selects departure/arrival states that are consistent with the best thrust/time trade.
- Sensitivity of t_{esc} and t_{cap} to spacecraft mass is smoother, improving SQP robustness.
- Overall propellant consumption is reduced by up to 2–3% compared to fixed-T Stage 1 maps.

CHAPTER 4. Stage 3: Optimal LEO-to-LLO Transfer

We now assemble the full LEO→LLO, spiral–coast–spiral, minimum-fuel problem in a single optimization, explicitly treating the thrust magnitude T on each powered arc as a free parameter within engine limits.

1. State and Control Variables

$$x(t) = \begin{pmatrix} r \\ v_r \\ v_\theta \\ \theta \\ m \end{pmatrix}, \qquad u(t) = \begin{pmatrix} T \\ \alpha \end{pmatrix}, \quad 0 \leq T \leq T_{\max}.$$

Here r, v_r, v_θ, θ are polar-frame position and velocity, and m is mass.

Due to the high sensitivity of reaching a precise lunar circular orbit, the LEO-to-LLO trajectory is split into two segments:

- **First Segment**: Begins in LEO, combining the powered Earth-escape spiral with part of the translunar coast.
- **Second Segment**: Starts in LLO, consisting of the powered lunar-capture spiral and the remaining coast phase.

To mitigate numerical challenges, the segments are matched near the Sphere of Influence (SOI), where Earth's and the Moon's gravitational forces are balanced.

A hybrid direct/indirect optimization technique is employed, blending features from optimal-control theory. The problem is formulated as follows: For a free end-time scenario, determine the initial angular position $\theta_1(0)$, thrust direction histories $u_1(t)$ (for $0 \leq t \leq t_{\text{escape}}$) and $u_2(t)$ (for $t_f - t_{\text{capture}} \leq t \leq t_f$), and thrust durations t_{escape} and t_{capture} to minimize:

$$J = \int_{t_0}^{t_f} \frac{T}{m} dt$$

2. Equations of Motion

The dynamics are governed by the following equations in an Earth-centered rotating frame:

$$\dot{r}_1 = v_{r_1} \tag{4.1}$$

$$\dot{v}_{r_1} = \frac{v_{\theta_1}^2}{r_1} - \frac{\mu_1}{r_1^2} - \frac{\mu_2(r_1 + D\cos\theta_1)}{r_{\text{moon-S/C}}^3} + \frac{\mu_2 \cos\theta_1}{D^2} + a_{T_1}\sin u_1 + 2\omega v_{\theta_1} + \omega^2 r_1 \tag{4.2}$$

$$\dot{v}_{\theta_1} = \frac{\mu_2 D \sin\theta_1}{r_{\text{moon-S/C}}^3} - \frac{\mu_2 \sin\theta_1}{D^2} + a_{T_1}\cos u_1 - 2\omega v_{r_1} - \frac{v_{r_1} v_{\theta_1}}{r_1} \tag{4.3}$$

$$\dot{\theta}_1 = \frac{v_{\theta_1}}{r_1} \tag{4.4}$$

$$\dot{m} = -\frac{T}{c} \tag{4.5}$$

where:

$$r_{\text{moon-S/C}} = \sqrt{r_1^2 + 2Dr_1\cos\theta_1 + D^2}, \quad 0 \leq T \leq T_{max}, \quad 0 \leq t \leq t_{\text{escape}}$$

For the moon-centered frame, the equations are:

$$\dot{r}_2 = -v_{r_2}$$

$$\dot{v}_{r_2} = -\left[\frac{v_{\theta_2}^2}{r_2} - \frac{\mu_2}{r_2^2} - \frac{\mu_1(r_2 - D\cos\theta_2)}{r_{\text{Earth-S/C}}^3} - \frac{\mu_1\cos\theta_2}{D^2} + a_{T_2}\sin u_2 + 2\omega v_{\theta_2} + \omega^2 r_2\right]$$

$$\dot{v}_{\theta_2} = -\left[-\frac{\mu_1 D\sin\theta_2}{r_{\text{Earth-S/C}}^3} + \frac{\mu_1\sin\theta_2}{D^2} + a_{T_2}\cos u_2 - 2\omega v_{r_2} - \frac{v_{r_2} v_{\theta_2}}{r_2}\right]$$

$$\dot{\theta}_2 = -\frac{v_{\theta_2}}{r_2}$$

$$\dot{m} = -\frac{T}{c}$$

where:

$$r_{\text{Earth-S/C}} = \sqrt{r_2^2 - 2Dr_2\cos\theta_2 + D^2}, \quad 0 \leq T \leq T_{max}, \quad t_f - t_{\text{capture}} \leq t \leq t_f$$

4.0.1 Boundary Conditions

Initial conditions for LEO:

$$\begin{aligned} r_1(0) &= r_{\text{LEO}} \\ v_{r_1}(0) &= 0 \\ v_{\theta_1}(0) &= \sqrt{\frac{\mu_1}{r_{\text{LEO}}}} - \omega r_{\text{LEO}} \\ m(0) &= m_{LEO} \end{aligned}$$

Terminal conditions for LLO:

$$\psi[x(t_f), t_f] = \begin{pmatrix} r_2(t_f) - r_{\text{LLO}} \\ v_{r_2}(t_f) \\ v_{\theta_2}(t_f) - \sqrt{\frac{\mu_2}{r_{\text{LLO}}}} - \omega r_{\text{LLO}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

The first four equations describe the spacecraft's motion in an Earth-centered rotating frame, while the next four represent the moon-centered frame, integrated backward in time. The initial and final conditions correspond to circular parking orbits around Earth and the Moon, respectively.

3. Objective

Minimize total propellant usage, equivalently

$$J = \int_0^{t_{\text{esc}}} \frac{T}{I_{\text{sp}} g_0}\, dt + \int_{t_{\text{esc}}+t_c}^{t_f} \frac{T}{I_{\text{sp}} g_0}\, dt,$$

where t_{esc}, t_c, t_{cap} are the durations of the escape spiral, coast, and capture spiral, and $t_f = t_{\text{esc}} + t_c + t_{\text{cap}}$.

4. Boundary and Continuity Conditions

- At $t = 0$: $r = r_{\text{LEO}},\ v_r = 0,\ v_\theta = \sqrt{\mu_E / r_{\text{LEO}}},\ \theta = 0,\ m = m_0$.
- Between stages: $\big(r, v_r, v_\theta, m\big)$ continuity at $t = t_{\text{esc}}$ and $t = t_{\text{esc}} + t_c$; coast-to-capture match enforced via Stage 1 curve-fits.

- At $t = t_f$: $r = r_{\text{LLO}},\ v_r = 0,\ v_\theta = \sqrt{\mu_M / r_{\text{LLO}}},\ \theta$ free, m free.
- Transversality: costates satisfy free-angle and tangent-steering conditions at each thrust endpoint.

5. Control Parameterization

Each thrust arc is discretized into N segments with constant (T_j, α_j). Thus

$$u(t) = \{(T_j, \alpha_j)\}_{j=1}^{N},$$

and T_j appear as optimization variables bounded by $0, T_{\max}$.

6. Numerical Solution

1. **Initial Guess:**
    - $t_{\text{esc}}, t_{\text{cap}}$ from Stage 2 coast end.
2. **NLP Formulation:** Minimize J subject to dynamics (via collocation), boundary and continuity constraints, and $0 \le T_j \le T_{\max}$.
3. **Solver:** Apply a direct SQP solver with analytic or finite-difference derivatives. However, the Thrust magnitude and direction profiles used are derived from the necessary conditions of optimality (indirect method). The free thrust variables T_j allow the optimizer to trade thrust level against burn time for improved propellant efficiency.
4. **Refinement:** Upon convergence, refine the collocation mesh and re-solve to ensure accuracy of the bang–bang / singular switching structure in $T(t)$.

CHAPTER 5. Numerical Example and Simulation Results

5.1 Retrograde Solution Performance

First, the retrograde trajectory was optimized for the constant thrust magnitude using the hybrid direct/indirect method, achieving a final spacecraft mass of **93,032 kg** in Low Lunar Orbit (LLO), corresponding to a mass ratio of **0.9303**. This marginally improved upon the suboptimal retrograde solution (93,025 kg). Key metrics include:

- **Match-point errors**: 1.9 lunar radii (position), 4.6°(angle), and velocity deviations of 0.07 km/s (radial) and 0.13 km/s (circumferential).
- **Trip time**: 7.36 days, with a 4.56-day translunar coast and a 12.72-hour lunar capture spiral.
- **Convergence**: Achieved in 16 iterations, demonstrating robustness of the hybrid method.

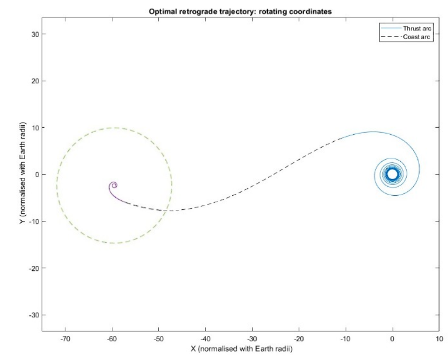

Figure 5.1: Optimal retrograde trajectory in Earth-centered rotating frame with Constant Thrust Magnitude.

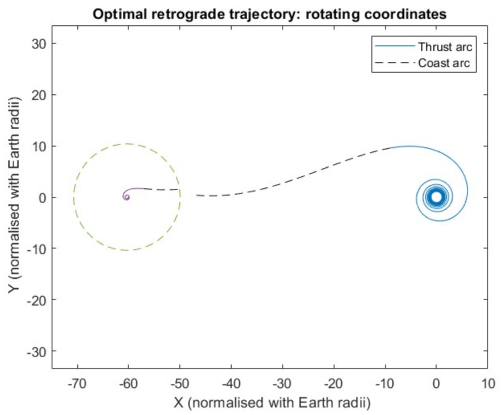

Figure 5.2: Optimal retrograde trajectory in Earth-centered rotating frame with Variable Thrust Magnitude.

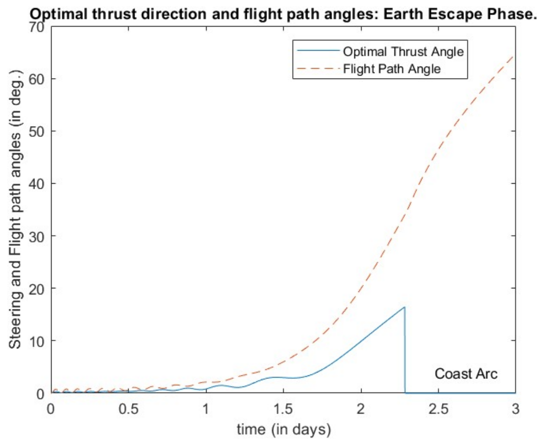

Figure 5.3: Optimal-thrust direction and flight path angles for Constant Thrust Magnitude: Earth escape

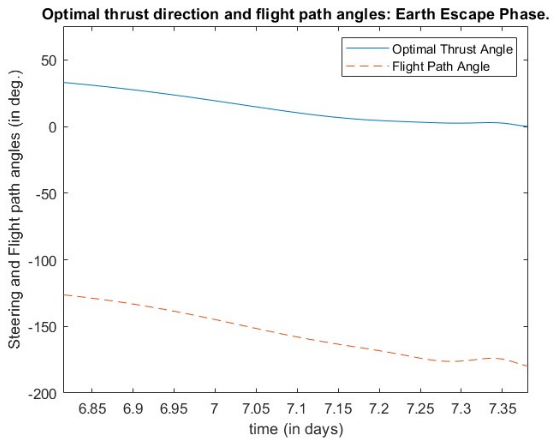

Figure 5.4: Optimal-thrust direction and flight path angles for Constant Thrust Magnitude: Moon-Capture

5.2 Comparison with Variable Thrust Formulation

Next, the variable thrust magnitude formulation was tested to explore potential fuel savings. However, this approach **failed to converge to a feasible solution**. Suspected causes include:

5.2.1 Suboptimal Initial Guesses

- The maximum-energy spiral assumption may not be a good functional to be used in the initial guess formulation for the variable thrust magnitude.
- *Planned mitigation*: Introduce a penalty function for fuel consumption during the initial guess phase to better guide the optimizer.

5.2.2 Code Debugging Challenges

- The hybrid method's equations (e.g., augmented state dynamics, costate transitions) are highly nonlinear.
- Potential issues include mismatched boundary conditions or numerical instabilities in the thrust-switching logic.

5.3 Future Work

To address non-convergence in the variable thrust case:

- **Refine initial guesses** using a composite objective (fuel + energy penalties).
- **Stepwise validation** of the hybrid method's components (e.g., costate propagation, thrust continuity checks).

CHAPTER 6. General Conclusion

This study tried to demonstrate the computation of minimum-fuel planar trajectories between Earth and lunar circular orbits for a low-thrust spacecraft with variable thrust magnitude with an initial thrust-to-weight ratio of 3×10^{-3}. The solution approach employed a structured three-stage methodology: first solving a two-body maximum-energy escape/capture problem, then a restricted three-body coasting problem, and finally the complete minimum-fuel transfer using an innovative hybrid direct/indirect optimization method. The results validate that the suboptimal coasting solution provides an excellent initial estimate for the full optimization problem, while the hybrid method effectively combines the robustness of direct optimization with the precision of indirect optimal control theory.

The three-body dynamics formulation proved particularly effective when handling the challenging terminal constraints for low lunar orbit. The Variable thrust magnitude formulation was explored as a potential enhancement to the fixed-thrust baseline solution. While theoretically promising for further fuel optimization, this approach presented significant convergence challenges in practice while strategically trying to match Earth-escape and moon-capture trajectory segments near the lunar sphere of influence. The difficulties likely stem from several factors: the increased complexity of the optimal control problem when thrust magnitude becomes a free parameter, the inadequacy of maximum-energy spiral solutions as initial guesses for variable thrust scenarios, and numerical sensitivities in the augmented system dynamics. These computational hurdles suggest that future work should focus on developing specialized initialization techniques, possibly incorporating fuel penalty terms or alternative guidance laws, to make the variable thrust optimization more tractable.

References

1. Pierson, B.L., Kluever, C.A. (1994). Three-stage approach to optimal low-thrust Earth Moon trajectories. Journal of Guidance, Control, and Dynamics, 17(6), 1275–1282. https://doi.org/10.2514/3.21344

2. Kluever, C.A., Pierson, B.L. (1995). Optimal low-thrust three-dimensional Earth–Moon trajectories. Journal of Guidance, Control, and Dynamics, 18(4), 830–837. https://doi.org/10.2514/3.21466

3. Kechichian, J.A. (1995). Optimal low-thrust transfer using variable bounded thrust. Acta Astronautica, 36(7), 357–365. https://doi.org/10.1016/0094-5765(95)00112-3

4. Dynamical Systems, the Three-Body Problem and Space Mission Design. Wang Sang Koon, Martin W. Lo, Jerrold E. Marsden, and Shane D. Ross. September 2000, 1167-1181

5. Systematic Design of Optimal Low-Thrust Transfers for the Three-Body Problem. Shankar Kulumani, Taeyoung Lee. AAS Astrodynamics Specialist Conference Vail, CO August 2015

6. Minimum fuel control of the planar circular restricted three-body problem. J.-B. Caillau · B. Daoud · J. Gergaud. Celest Mech Dyn Astr (2012) 114:137–150.

7. Minimum time control of the restricted three-body problem. Jean-Baptiste Caillau, Bilel Daoud. 2011.

8. The optimal control of the circular restricted three body problem. Bilel Daoud. Optimization and Control. Université de Bourgogne, 2011. English.

APPENDIX A. MATLAB Code

```
%% This script is for computing the thrust angle to maximize the spacecraft
% energy at the end of a fixed-end-time escape spiral from LEO.

clc;
clear;
close all;

%% Input
mu_e = 3.986004418*1e14; % in m^3/s^-2
g = 9.80665; % standard gravity acceleration at sea level (in m/s^2)
Re = earthRadius; % Radius of earth (in m)
mdot = 107.5/3600; % time rate of change of mass due to thrust (in kg/s)

%Isp = 10046.5287022250; % Constant Specific Impulse value
Isp = 700; % Constant Specific Impulse value
T_max = 2942;
%T_max = 3542;
c = Isp*g; % Exhaust velocity (in m/s)

m_leo = 1e5; % initial mass at LEO (in kg)
r_leo = Re + 315*1e3; % initial orbit radius (in m)
v_theta_leo = sqrt(mu_e/r_leo);
```

```
% Define time vector
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds
t1 = 1.67*t_c; % Final escape time vector in seconds
%t1 = 197285.182393481;

% Initial conditions (from Equations of Motion)
%x0 = zeros(4,1);
x0 = zeros(5,1);

x0(1) = r_leo;
x0(3) = v_theta_leo;
x0(5) = m_leo;

%% Solve (Using fmincon)
% Initial guess for missing initial conditions or parameters
%s0 = [-mu_e/r_leo^2; 0; -v_theta_leo];
s0 = [-mu_e/r_leo^2; -1; -v_theta_leo; -10];

% Optimization options for fmincon (SQP)
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'TolFun', 1e-10);

% Optimize using fmincon
s_solution = fmincon(@(s) Escape_Cost_Function(s, x0, t0, t1), s0, [], [], [], [],

%% Solution
% Solve ODEs using the correct initial conditions
```

```
options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-10);
%[t, y] = ode45(@(t,y) Escape_ODE(t, y), [t0, t1], initial_conditions(s_solution, s

[t, y] = ode45(@(t,y) VariableThrustODE(t, y), [t0, t1], initial_conditions(s_solut

%% Data for testing curve fits
rf = y(end, 1);
vr_f = y(end, 2);
vtheta_f = y(end, 3);
mf = m_leo - mdot*t1;
J = (vr_f^2 + vtheta_f^2)/2 - mu_e/rf;
%% Plot Trajectory and Optimal Thrust angle Profile
r = y(:,1);
theta = y(:,4);

% Position in Cartesian Coordinates
x_vec = r.*cos(theta);
y_vec = r.*sin(theta);

% Optimal Control
%u = atan2(-y(:,6),-y(:,7)); % Optimal control
u = atan2(-y(:,7),-y(:,8)); % Optimal control
gam = atan2(y(:,2), y(:,3)); % Flight path angle

figure;
plot(x_vec/Re,y_vec/Re)
```

```
axis('equal')
xlabel(' X (normalised with Earth radii)')
ylabel(' Y (normalised with Earth radii)')
title('Maximum-energy Earth-escape spiral for t_f = 2.37 days')

figure;
plot(t/t_c, u*180/pi)
xlabel('time (in days)')
ylabel('Steering and Flight path angles (in deg.)')
hold on
plot(t/t_c, gam*180/pi, '--')
legend('Optimal Thrust Angle', 'Flight Path Angle')
title('Optimal thrust direction and flight path angles: Earth Escape Phase.')

%% This script is for computing the thrust angle to maximize the spacecraft
% energy at the end of start of a fixed-time capture spiral in LLO.

clc;
clear;
close all;

global m_c;

%% Input
mu_m = 4.9048695*1e12; % in m^3/s^-2
g = 9.80665; % standard gravity acceleration at sea level (in m/s^2)
Rm = 1737.1*1e3; % Radius of moon (in m)
```

```
mdot = -107.5/3600; % time rate of change of mass due to thrust (in kg/s)
%m_llo = 0.89*1e5; % mass left at LLO (in kg)
m_llo = 92990.1164250887; % mass left at LLO (in kg)

%Isp = 10046.5287022250; % Constant Specific Impulse value
%T_max = 2942;
%c = Isp*g; % Exhaust velocity (in m/s)

r_llo = Rm + 100*1e3; % initial orbit radius (in m)
v_theta_llo = -sqrt(mu_m/r_llo);

% Define time vector
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds
%t1 = 15.6*60*60; % Final escape time vector in seconds
t1 = 40502.1515870877;

m_c = m_llo; % mass left at LL0 (in kg)

% Initial conditions (from Equations of Motion)
x0 = zeros(4,1);
%x0 = zeros(5,1);

x0(1) = r_llo;
x0(3) = v_theta_llo;
%x0(5) = m_leo;
```

```
%% Solve (Using fmincon)
% Initial guess for missing initial conditions or parameters
s0 = [-mu_m/r_llo^2; 0; -v_theta_llo];
%s0 = [-mu_e/r_leo^2; 10; -v_theta_leo; (T_max/m_leo)*v_theta_leo];

% Optimization options for fmincon (SQP)
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'TolFun', 1e-8);

% Optimize using fmincon
s_solution = fmincon(@(s) Capture_Cost_Function(s, x0, t0, t1), s0, [], [], [], [],

%% Solution
% Solve ODEs using the correct initial conditions
options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[t, y] = ode45(@(t,y) Capture_ODE(t, y), [t0, t1], initial_conditions(s_solution, x

%[t, y] = ode45(@(t,y) VariableThrustODE(t, y), [0, t1], initial_conditions(s_solut

%% Data for testing curve fits
rf = y(end, 1);
vr_f = y(end, 2);
vtheta_f = y(end, 3);
mf = m_llo - mdot*t1;
J = (vr_f^2 + vtheta_f^2)/2 - mu_m/rf;
%% Plot Trajectory and Optimal Thrust angle Profile
r = y(:,1);
theta = y(:,4);
```

```
% Position in Cartesian Coordinates
x_vec = r.*cos(theta);
y_vec = r.*sin(theta);

% Optimal Control
u = atan2(-y(:,6),-y(:,7))+pi; % Optimal control
%u = atan2(-y(:,7),-y(:,8)); % Optimal control
gam = atan2(y(:,2), y(:,3)); % Flight path angle

figure;
plot(x_vec/Rm, y_vec/Rm)

axis('equal')
xlabel(' X (normalised with Earth radii)')
ylabel(' Y (normalised with Earth radii)')
title('Maximum-energy Earth-escape spiral for t_f = 2.37 days')

figure;
plot(t/t_c, u*180/pi)
xlabel('time (in days)')
ylabel('Steering and Flight path angles (in deg.)')
hold on
plot(t/t_c, gam*180/pi, '--')
legend('Optimal Thrust Angle', 'Flight Path Angle')
title('Optimal thrust direction and flight path angles: Earth Escape Phase.')
```

```
function J = Escape_Cost_Function(s, x0, t0, t1)
mu_e = 3.986004418*1e14; % in m^3/s^-2

% Solve ODEs using the guessed parameters
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
%[~, y] = ode45(@(t,y) Escape_ODE(t, y), [t0, t1], initial_conditions(s, x0), optic

[~, y] = ode45(@(t,y) VariableThrustODE(t, y), [t0, t1], initial_conditions(s, x0)


% Define the cost as the sum of squared errors in terminal conditions
J = 0;

%% Constraint Function (Ensures Boundary Conditions are Satisfied)
function [cineq, ceq] = Escape_Constraint_Function(s, x0, t0, t1)
mu_e = 3.986004418*1e14; % in m^3/s^-2

% Solve ODEs using the guessed parameters
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
%[~, y] = ode45(@(t,y) Escape_ODE(t, y), [t0, t1], initial_conditions(s, x0), optic
[~, y] = ode45(@(t,y) VariableThrustODE(t, y), [t0, t1], initial_conditions(s, x0)

% Extract final values at t1
y_final = y(end, :);

% No inequality constraints (c = [])
cineq = [];
```

```
% Equality constraints (terminal conditions)
%ceq = [y_final(5) + mu_e/(y_final(1)^2); y_final(6) + y_final(2); y_final(7) + y_

%ceq = [y_final(6) + mu_e/(y_final(1)^2); y_final(7) + y_final(2); y_final(8) + y_
ceq = [y_final(6) + mu_e/(y_final(1)^2); y_final(7) + y_final(2); y_final(8) + y_fi

function dydt = Escape_ODE(t, y)

mu_e = 3.986004418*1e14; % in m^3/s^-2

T = 2942; % Constant Thrust magnitude (in N)
m_leo = 1e5; % initial mass at LEO (in kg)
mdot = 107.5/3600; % time rate of change of mass due to thrust (in kg/s)

m = m_leo - mdot*t;
a_T = T/m;

% Define your system of 7 differential equations
dydt = zeros(7,1);

u = atan2(-y(6), -y(7)); % Optimal control

%For escape
dydt(1) = y(2);
dydt(2) = y(3)^2/y(1) - mu_e/y(1)^2 + a_T*sin(u);
dydt(3) = -y(2)*y(3)/y(1) + a_T*cos(u);
dydt(4) = y(3)/y(1);
dydt(5) = y(6)*(y(3)^2/y(1)^2 - 2*mu_e/y(1)^3) - y(7)*y(2)*y(3)/(y(1)^2);
dydt(6) = -y(5) + y(7)*y(3)/y(1);
dydt(7) = -2*y(6)*y(3)/y(1) + y(7)*y(2)/y(1);

function dydt = VariableThrustODE(t, y)


mu_e = 3.986004418*1e14; % in m^3/s^-2
g = 9.80665;
%Isp = 10046.5287022250; % Constant Specific Impulse value
Isp = 700;
%T_max = 2942;
T_max = 2942;
c = Isp*g; % Exhaust velocity (in m/s)
%m_leo = 1e5; % initial mass at LEO (in kg)

alph = atan2(-y(7),-y(8)); % Optimal Thrust angle
X = (y(7)*sin(alph) + y(8)*cos(alph))/y(5) - y(9)/c;

if X < 0
    T = T_max;
elseif X > 0
    T = 0;
%else
%    T = (c^2/(y(9)*(c-1)))*((-y(7)+y(8)*y(3)/y(1))*sin(alph) + (-2*y(7)*y(3)/y(1)
end
```

```
disp(X);

a_T = T/y(5);

% Define your system of 7 differential equations
dydt = zeros(9,1);

dydt(1) = y(2);
dydt(2) = y(3)^2/y(1) - mu_e/y(1)^2 + a_T*sin(alph);
dydt(3) = -y(2)*y(3)/y(1) + a_T*cos(alph);
dydt(4) = y(3)/y(1);
dydt(5) = -T/c;
dydt(6) = y(7)*(y(3)^2/y(1)^2 - 2*mu_e/y(1)^3) - y(8)*(y(2)*y(3)/y(1)^2);
dydt(7) = -y(6) + y(8)*y(3)/y(1);
dydt(8) = -2*y(7)*y(3)/y(1) + y(8)*y(2)/y(1);
dydt(9) = (a_T/y(5))*(y(7)*sin(alph) + y(8)*cos(alph));

function y0 = initial_conditions(s, x0)
% Construct the full initial condition vector
y0 = [x0; s];

%%% This script is to reproduce the results from the paper titled
% "Three-stage approach to optimal low-thrust Earth-moon trajectories."

clc;
clear;
close all;
```

```
%% Input
mu_e = 3.986004418*1e14; % in m^3/s^-2
g = 9.80665; % standard gravity acceleration at sea level (in m/s^2)
Re = earthRadius; % Radius of earth (in m)

m_leo = 1e5; % initial mass at LEO (in kg)
r_leo = Re + 315*1e3; % initial orbit radius (in m)
v_theta_leo = sqrt(mu_e/r_leo);

% Define time vector
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds
t1_vec = linspace(2.1, 2.37, 6)*t_c; % Final escape time vector in seconds
dt = 100; % in seconds

% Initial conditions (from Equations of Motion)
x0 = zeros(4,1);

x0(1) = r_leo;
x0(3) = v_theta_leo;

%% Curve Fitting (for Earth-Escape Maximum Spiral Trajectory
vr_vec = zeros(1, length(t1_vec)); % Final radial velocity vector
vtheta_vec = zeros(1, length(t1_vec)); % Final tangential velocity vector
rf_vec = zeros(1, length(t1_vec)); % Final radius vector
```

```
for i = 1:length(t1_vec)
    t1 = t1_vec(i);
    %% Solve (Using fmincon)
    % Initial guess for missing initial conditions or parameters
    s0 = [-mu_e/r_leo^2; 0; -v_theta_leo];

    % Optimization options for fmincon (SQP)
    options = optimoptions('fmincon', 'Algorithm', 'sqp', 'TolFun', 1e-8);

    % Optimize using fmincon
    s_solution = fmincon(@(s) Escape_Cost_Function(s, x0, t0, t1), s0, [], [], [],

    %% Solution
    % Solve ODEs using the correct initial conditions
    options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
    [t, y] = ode45(@(t,y) Escape_ODE(t, y), [0, t1], initial_conditions(s_solution

    %% Update vectors (for curve fitting)
    rf_vec(i) = y(end, 1);
    vr_vec(i) = y(end, 2);
    vtheta_vec(i) = y(end, 3);
end

%% Curve fit the following functions
% Choose a degree n
n = 5;
```

```
% Fit polynomials
p_vr = polyfit(rf_vec, vr_vec, n);          % vr = f1(rf)
p_vtheta = polyfit(rf_vec, vtheta_vec, n); % vtheta = f2(rf)
p_tf = polyfit(rf_vec, t1_vec, n);          % tf = f3(rf)

% Evaluate at a new rf (e.g., rf_new)
rf_new = 65562185.8252543;  % example value
vr_new = polyval(p_vr, rf_new);
vtheta_new = polyval(p_vtheta, rf_new);
tf_new = polyval(p_tf, rf_new);

% Save the polynomial coefficients to a .mat file
save('Escape_Curve_Fitting_Function.mat', 'p_vr', 'p_vtheta', 'p_tf');

%% Constraint Function (Ensures Boundary Conditions are Satisfied)
function [c, ceq] = Capture_Constraint_Function(s, x0, t0, t1)
mu_m = 4.9048695*1e12; % in m^3/s^-2

% Solve ODEs using the guessed parameters
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[~, y] = ode45(@(t,y) Capture_ODE(t, y), [t0, t1], initial_conditions(s, x0), optic
%[~, y] = ode45(@(t,y) VariableThrustODE(t, y), [t0, t1], initial_conditions(s, x0,

% Extract final values at t1
y_final = y(end, :);

% No inequality constraints (c = [])
```

```
c = [];

% Equality constraints (terminal conditions)
ceq = [y_final(5) + mu_m/y_final(1)^2; y_final(6) + y_final(2); y_final(7) + y_fina

function J = Capture_Cost_Function(s, x0, t0, t1)

mu_m = 4.9048695*1e12; % in m^3/s^-2

%g = 9.80665;
%Isp = 10046.5287022250; % Constant Specific Impulse value
%c = Isp*g; % Exhaust velocity (in m/s)
%T_max = 2942;

% Solve ODEs using the guessed parameters
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[~, y] = ode45(@(t,y) Capture_ODE(t, y), [t0, t1], initial_conditions(s, x0), optio

%[~, y] = ode45(@(t,y) VariableThrustODE(t, y), [t0, t1], initial_conditions(s, x0,


% Define the cost as the sum of squared errors in terminal conditions
J = 0;

%% Define the cost as the sum of squared errors in terminal conditions (Variable T
% alph_final = atan2(-y_final(7),-y_final(8)); % Optimal Thrust angle
% %m_inv_final = (y_final(9)/c)*(1/(y_final(7)*sin(alph_final) + y_final(8)*cos(al
% X_final = (y_final(7)*sin(alph_final) + y_final(8)*cos(alph_final))/y_final(5) -
%
% if X_final < 0
%       T_final = T_max;
% elseif X_final > 0
%       T_final = 0;
% else
%       T_final = (c^2/(y_final(9)*(c-1)))*((-y_final(7)+y_final(8)*y_final(3)/y_fin
% end
%
% J = (y_final(6) + mu_e/y_final(1)^2)^2 + (y_final(9) + (T_final/y_final(5)^2)*(y

function dydt = Capture_ODE(t, y)
global m_c;

mu_m = 4.9048695*1e12; % in m^3/s^-2

T = 2942; % Constant Thrust magnitude (in N)
mdot = -107.5/3600; % time rate of change of mass due to thrust (in kg/s)

m = m_c - mdot*t;
a_T = T/m;

% Define your system of 7 differential equations
dydt = zeros(7,1);

u = atan2(-y(6), -y(7))+pi; % Optimal control
```

```
%for capture
dydt(1) = -y(2);
dydt(2) = -(y(3)^2/y(1) - mu_m/y(1)^2 + a_T*sin(u));
dydt(3) = -(-y(2)*y(3)/y(1) + a_T*cos(u));
dydt(4) = -y(3)/y(1);
dydt(5) = -(y(6)*(y(3)^2/y(1)^2 - 2*mu_m/y(1)^3) - y(7)*y(2)*y(3)/(y(1)^2));
dydt(6) = -(-y(5) + y(7)*y(3)/y(1));
dydt(7) = -(-2*y(6)*y(3)/y(1) + y(7)*y(2)/y(1));

%%% This script is to reproduce the results from the paper titled
% "Three-stage approach to optimal low-thrust Earth-moon trajectories."

clc;
clear;
close all;

global m_c;

%% Input
mu_m = 4.9048695*1e12; % in m^3/s^-2
g = 9.80665; % standard gravity acceleration at sea level (in m/s^2)
Rm = 1737.1*1e3; % Radius of moon (in m)
mdot = -107.5/3600; % time rate of change of mass due to thrust (in kg/s)

%Isp = 10046.5287022250; % Constant Specific Impulse value
%T_max = 2942;
```

```
%c = Isp*g; % Exhaust velocity (in m/s)

r_llo = Rm + 100*1e3; % initial orbit radius (in m)
v_theta_llo = sqrt(mu_m/r_llo);

% Define time vector
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds

% Initial conditions (from Equations of Motion)
x0 = zeros(4,1);
x0(1) = r_llo;
x0(3) = v_theta_llo;

%% Curve Fitting (for Earth-Escape Maximum Spiral Trajectory
m_llo_vec = linspace(0.86, 0.93, 4)*1e5; % vector of mass left at LLO (in kg)
rf_vec = linspace(7, 30, 5)*Rm; % Final radius vector vector at start of capture (

%m_llo_vec = 0.86*1e5;
%rf_vec = 7*Rm;

vr_vec = zeros(length(m_llo_vec), length(rf_vec)); % Final radial velocity vector
vtheta_vec = zeros(length(m_llo_vec), length(rf_vec)); % Final tangential velocity
t1_vec = zeros(length(m_llo_vec), length(rf_vec)); % Capture time vector

itr_max = 1000; % max number of iterations for convergence
```

```
for i = 1:length(m_llo_vec)
    m_c = m_llo_vec(i);
    for j = 1:length(rf_vec)
        t1 = 15.6*3600; % intial guess for t1 (in s)
        for k = 1:itr_max
            disp(k)
            %% Solve (Using fmincon)
            % Initial guess for missing initial conditions or parameters
            s0 = [-mu_m/r_llo^2; 0; -v_theta_llo];

            % Optimization options for fmincon (SQP)
            options = optimoptions('fmincon', 'Algorithm', 'sqp', 'TolFun', 1e-8);

            % Optimize using fmincon
            s_solution = fmincon(@(s) Capture_Cost_Function(s, x0, t0, t1), s0, [],

            %% Solution
            % Solve ODEs using the correct initial conditions
            options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
            [t, y] = ode45(@(t,y) Capture_ODE(t, y), [t0, t1], initial_conditions(s

            %[t, y] = ode45(@(t,y) VariableThrustODE(t, y), [0, t1], initial_condit

            %% Newton's Method
            rf = y(end, 1);
            rf_dot = y(end, 2);
            vtheta_f = y(end, 3);
            del_t1 = (rf_vec(j) - rf)/rf_dot;
            if abs(del_t1) < 0.02
                t1_vec(i,j) = t1;
                vr_vec(i,j) = rf_dot;
                vtheta_vec(i,j) = vtheta_f;
                break;
            end

            % update step
            t1 = t1-del_t1;
        end

    end
end



%% Curve fit the following functions
% Evaluate at a specific point
rf_query = 680016782.639722;
m_llo_query = 0.94*1e5;

% Create the spline interpolants
[vr_spline, vtheta_spline, tf_spline] = Capture_Splines(rf_vec, m_llo_vec, t1_vec,

vr_val = vr_spline(rf_query, m_llo_query);
vtheta_val = vtheta_spline(rf_query, m_llo_query);
```

```
tf_val = tf_spline(rf_query, m_llo_query);

% Save the polynomial coefficients to a .mat file
save('Capture_Spline_Function.mat', 'vr_spline', 'vtheta_spline', 'tf_spline');

function [vr_spline, vtheta_spline, tf_spline] = Capture_Splines(rf_vec, m_llo_vec,
    % CREATE_VELOCITY_SPLINES Creates 2D cubic spline interpolants
    %
    % Input organization:
    %   tf_vec, vr_vec, vtheta_vec are 2D arrays with:
    %   - rows corresponding to m_llo_vec (spacecraft mass)
    %   - columns corresponding to rf_vec (moon radial distance)
    %
    % Outputs:
    %   Three griddedInterpolant objects in NDGRID format

    % Verify input dimensions
    [n_rows, n_cols] = size(tf_vec);
    if ~isequal(size(vr_vec), [n_rows, n_cols]) || ~isequal(size(vtheta_vec), [n_ro
        error('All input 2D arrays must have the same dimensions');
    end

    if length(m_llo_vec) ~= n_rows || length(rf_vec) ~= n_cols
        error('Dimension mismatch: length(m_llo_vec) must match rows, length(rf_vec
    end

    % Convert to NDGRID format that griddedInterpolant expects
    % Note the order of inputs to ndgrid matches your data organization
    [rf_grid_nd, m_llo_grid_nd] = ndgrid(rf_vec, m_llo_vec);

    % Create the interpolants - transposing the data arrays is CRUCIAL
    vr_spline = griddedInterpolant(rf_grid_nd, m_llo_grid_nd, vr_vec', 'spline');
    vtheta_spline = griddedInterpolant(rf_grid_nd, m_llo_grid_nd, vtheta_vec', 'spl
    tf_spline = griddedInterpolant(rf_grid_nd, m_llo_grid_nd, tf_vec', 'spline');
end

%% This script is for computing the thrust angle to maximize the spacecraft
% energy at the end of start of a fixed-time capture spiral in LLO.

clc;
clear;
close all;

global m_c;

%% Input
mu_m = 4.9048695*1e12; % in m^3/s^-2
g = 9.80665; % standard gravity acceleration at sea level (in m/s^2)
Rm = 1737.1*1e3; % Radius of moon (in m)
mdot = -107.5/3600; % time rate of change of mass due to thrust (in kg/s)
%m_llo = 0.89*1e5; % mass left at LLO (in kg)
m_llo = 92990.1164250887; % mass left at LLO (in kg)

%Isp = 10046.5287022250; % Constant Specific Impulse value
```

```
%T_max = 2942;
%c = Isp*g; % Exhaust velocity (in m/s)

r_llo = Rm + 100*1e3; % initial orbit radius (in m)
v_theta_llo = -sqrt(mu_m/r_llo);

% Define time vector
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds
%t1 = 15.6*60*60; % Final escape time vector in seconds
t1 = 40502.1515870877;

m_c = m_llo; % mass left at LL0 (in kg)

% Initial conditions (from Equations of Motion)
x0 = zeros(4,1);
%x0 = zeros(5,1);

x0(1) = r_llo;
x0(3) = v_theta_llo;
%x0(5) = m_leo;

%% Solve (Using fmincon)
% Initial guess for missing initial conditions or parameters
s0 = [-mu_m/r_llo^2; 0; -v_theta_llo];
%s0 = [-mu_e/r_leo^2; 10; -v_theta_leo; (T_max/m_leo)*v_theta_leo];
```

```
% Optimization options for fmincon (SQP)
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'TolFun', 1e-8);

% Optimize using fmincon
s_solution = fmincon(@(s) Capture_Cost_Function(s, x0, t0, t1), s0, [], [], [], [],

%% Solution
% Solve ODEs using the correct initial conditions
options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[t, y] = ode45(@(t,y) Capture_ODE(t, y), [t0, t1], initial_conditions(s_solution, x

%[t, y] = ode45(@(t,y) VariableThrustODE(t, y), [0, t1], initial_conditions(s_solut

%% Data for testing curve fits
rf = y(end, 1);
vr_f = y(end, 2);
vtheta_f = y(end, 3);
mf = m_llo - mdot*t1;
J = (vr_f^2 + vtheta_f^2)/2 - mu_m/rf;
%% Plot Trajectory and Optimal Thrust angle Profile
r = y(:,1);
theta = y(:,4);

% Position in Cartesian Coordinates
x_vec = r.*cos(theta);
y_vec = r.*sin(theta);
```

```
% Optimal Control
u = atan2(-y(:,6),-y(:,7))+pi; % Optimal control
%u = atan2(-y(:,7),-y(:,8)); % Optimal control
gam = atan2(y(:,2), y(:,3)); % Flight path angle

figure;
plot(x_vec/Rm,y_vec/Rm)

axis('equal')
xlabel(' X (normalised with Earth radii)')
ylabel(' Y (normalised with Earth radii)')
title('Maximum-energy Earth-escape spiral for t_f = 2.37 days')

figure;
plot(t/t_c, u*180/pi)
xlabel('time (in days)')
ylabel('Steering and Flight path angles (in deg.)')
hold on
plot(t/t_c, gam*180/pi, '--')
legend('Optimal Thrust Angle', 'Flight Path Angle')
title('Optimal thrust direction and flight path angles: Earth Escape Phase.')

%% This script is for computing the trajectory for the coasting phase

clc;
clear;
close all;
```

```
% Load the polynomial coefficients from the .mat file
load('Escape_Curve_Fitting_Function.mat');
load('Capture_Spline_Function.mat');

%% Input
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu_e = 3.986004418*1e14; % in m^3/s^-2
Re = earthRadius; % Radius of earth (in m)
Rm = 1737.1*1e3; % Radius of moon (in m)
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system
m_leo = 1e5; % Initial mass at start of spiral
mdot = 107.5/3600; % time rate of change of mass due to thrust (in kg/s)
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds

%% Optimisation
% escape and capture curve fitting function
e_cf = [p_vr; p_vtheta; p_tf];

% Initial guess
%s0 = [9.69*Re; 50*pi/180; 0.93*m_leo; 3*t_c];
%s0 = [9.47*Re; 136*pi/180; 0.93*m_leo; 4.5*t_c];
```

```
%s0 = [11.29*Re; 10*pi/180; 0.87*m_leo; 1*t_c];
s0 = [8.29*Re; 105*pi/180; 0.96*m_leo; 5.2*t_c];



%lb = [6*Re, 0, 0.06*m_leo, 0.001*t_c]; % Lower bounds
%ub = [20*Re, 2*pi, 0.96*m_leo, 10*t_c];

% Optimization options for fmincon (SQP)
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'Display', 'iter', 'MaxFunct

% Optimize using fmincon
s_solution = fmincon(@(s) Coasting_Cost_Function(s, e_cf, vr_spline, vtheta_spline,

% options = optimoptions('patternsearch', ...
%     'Display', 'iter', ...            % Show iteration details
%     'MaxFunctionEvaluations', 10000, ... % Max function evaluations
%     'PlotFcn', {@psplotbestf, @psplotfuncount}); % Optional: Plot progress
%
% s_solution = patternsearch(@(s) Coasting_Cost_Function(s, e_cf, vr_spline, vthet

%% Simulation of Optimal Coast trajectory
r0 = s_solution(1); % intial radius at start of coast
theta0 = s_solution(2); % initial angle at start of coast
m_llo = s_solution(3); % Final mass in LLO (in kg)
t1 = s_solution(4); % duration of coast

vr_0 = polyval(p_vr, r0);
```

```
vtheta_0 = polyval(p_vtheta, r0) - omg*r0;

x0 = [r0; vr_0; vtheta_0; theta0];

% Solve ODEs using the guessed parameters
options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[~, y] = ode45(@(t,y) Coasting_ODE(t, y), [0, t1], x0, options_ode);

% Extract final values at t1
y_final = y(end, :);

rf = y_final(1);
vr_f = y_final(2);
vtheta_f = y_final(3);
theta_f = y_final(4);

[r2_f, theta2_f, vr2_f, vtheta2_f] = ECRF_to_MCRF(rf, theta_f, vr_f, vtheta_f, D, 

%% Optimal Solution Analysis
t_esc = polyval(p_tf, r0);
t_capt = tf_spline(r2_f, m_llo);

%% Plot Trajectory
r = y(:,1);
theta = y(:,4);

% Position in Cartesian Coordinates
```

```
x_vec = r.*cos(theta);
y_vec = r.*sin(theta);

figure;
plot(x_vec/Re, y_vec/Re)

axis('equal')
xlabel(' X (normalised with Earth radii)')
ylabel(' Y (normalised with Earth radii)')
title('Coasting Trajectory')



function dydt = Coasting_ODE(t, y)

% constant values for Earth-Moon system
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system

%% Set Thrust



% CR3BP EoM in Earth-centered polar rotating frame
dydt = zeros(4,1);
```

```
dydt(1) = y(2);
dydt(2) = y(3)^2/y(1) - mu_e/y(1)^2 - mu_m*(y(1) + D*cos(y(4)))/(y(1)^2 + 2*D*y(1)*
dydt(3) = mu_m*D*sin(y(4))/(y(1)^2 + 2*D*y(1)*cos(y(4)) + D^2)^(3/2) - mu_m*sin(y(
dydt(4) = y(3)/y(1);


function J = Coasting_Cost_Function(s, e_cf, vr_spline, vtheta_spline, tf_spline)

%% Constants
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system

%% Escape curve fitting functions
p_vr = e_cf(1, :);
p_vtheta = e_cf(2, :);
p_tf = e_cf(3, :);

%% Initial value for the EoM
r0 = s(1); % intial radius at start of coast
theta0 = s(2); % initial angle at start of coast
m_llo = s(3); % Final mass in LLO (in kg)
t1 = s(4); % duration of coast

vr_0 = polyval(p_vr, r0);
```

```
vtheta_0 = polyval(p_vtheta, r0) - omg*r0; % velocity in earth-centered rotating fr

x0 = [r0; vr_0; vtheta_0; theta0];

% Solve ODEs using the guessed parameters
options_ode = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[~, y] = ode45(@(t,y) Coasting_ODE(t, y), [0, t1], x0, options_ode);

% Extract final values at t1
y_final = y(end, :);
rf = y_final(1);
theta_f = y_final(4);

r2_f = sqrt(rf^2 + D^2 - 2*rf*D*cos(theta_f)); % radius at tf in moon-centered fram

% Compute escape and capture times
t_esc = polyval(p_tf, r0);
t_capt = tf_spline(r2_f, m_llo);

% Cost function
J = t_esc + t_capt;



%% Constraint Function (Ensures Boundary Conditions are Satisfied)
function [c, ceq] = Coasting_Constraint_Function(s, e_cf, vr_spline, vtheta_spline,

% Transform from Earth-Centered to Moon-centered
```

```
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system

%% Escape curve fitting functions
p_vr = e_cf(1, :);
p_vtheta = e_cf(2, :);
p_tf = e_cf(3, :);

%% Initial value for the EoM
r0 = s(1); % intial radius at start of coast
theta0 = s(2); % initial angle at start of coast
m_llo = s(3); % Final mass in LLO (in kg)
t1 = s(4); % duration of coast

vr_0 = polyval(p_vr, r0);
vtheta_0 = polyval(p_vtheta, r0) - omg*r0;

x0 = [r0; vr_0; vtheta_0; theta0];

%% Solve ODEs using the guessed parameters
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[~, y] = ode45(@(t,y) Coasting_ODE(t, y), [0, t1], x0, options);

% Extract final values at t1
```

```
y_final = y(end, :);
rf = y_final(1);
vr_f = y_final(2);
vtheta_f = y_final(3);
theta_f = y_final(4);

[r2_f, theta2_f, vr2_f, vtheta2_f] = ECRF_to_MCRF(rf, theta_f, vr_f, vtheta_f, D, o
% r2_f = sqrt(rf^2 + D^2 - 2*rf*D*cos(theta_f)); % radius at tf in moon-centered fr
% vr2_f = vr_f * cos(theta_f) + vtheta_f * sin(theta_f);
% vtheta2_f = -vr_f * sin(theta_f) + vtheta_f * cos(theta_f) - omg*r2_f;

% No inequality constraints (c = [])
c = [-r0; -t1];

% Compute escape and capture times
t_esc = polyval(p_tf, r0);
t_capt = tf_spline(r2_f, m_llo);

% Equality constraints (terminal conditions)
ceq = [vr2_f - vr_spline(r2_f, m_llo); abs(vtheta2_f) - abs(vtheta_spline(r2_f, m_



%% This script is for implementing the hybrid direct/indirect optimizer for
% generating optimal low-thrust earth-moon trajectory.

clc;
clear;
```

```
close all;

%% Input
Re = earthRadius; % Radius of earth (in m)
Rm = 1737.1*1e3; % Radius of moon (in m)
soi_m = 66100*1e3; % Moon's SOI radius (km)
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system
m_leo = 1e5; % Initial mass at start of spiral
mdot = 107.5/3600; % time rate of change of mass due to thrust (in kg/s)
t_c = 24*60*60; % constant to convert from days to seconds
t0 = 0*t_c; % in seconds

% LEO Data
r_leo = Re + 315*1e3; % initial orbit radius (in m)
v_theta_leo = sqrt(mu_e/r_leo);

% LLO Data
r_llo = Rm + 100*1e3; % initial orbit radius (in m)
v_theta_llo =sqrt(mu_m/r_llo);

%% Optimisation
x0_1 = [r_leo; 0; v_theta_leo - omg*r_leo]; % intial conditions for escape + partic
x0_2 = [r_llo; 0; v_theta_llo - omg*r_llo]; % intial conditions for escape + partic
```

```
theta_0 = 0.*pi/180;
theta_f = 0.2*pi/180;

% theta_0 = 2.2532;
% theta_f = -5.3835;

% theta_0 = -2.2532;
% theta_f = 0.1;

lam_0 = [-mu_e/r_leo^2; 0; -v_theta_leo];
lam_f = [-mu_m/r_llo^2; 0; -v_theta_llo];

t_esc0 = 197293.978080361;
t_capt0 = 49789.9037231618;
t_coast0 = 218388.470468064;

% t_esc0 = 192915.837887292;
% t_capt0 = 38657.0176538703;
% t_coast0 = 409697.479401245;

% t_esc0 = 193347.111945247;
% t_capt0 = 39753.4248502906;
% t_coast0 = 509369.174009751;

s0 = [theta_0; theta_f; lam_0; lam_f; t_esc0; t_capt0; t_coast0]; % parameters to
```

```
tmax = 3.5*t_c; % Initial maximum value for computing t_coast2

% Optimization options for fmincon (SQP) %'MaxFunctionEvaluations', 10000
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'Display', 'iter', 'StepToler

% Optimize using fmincon
% options = optimoptions('patternsearch', ...
%       'Display', 'iter', ...              % Show iteration details
%       'MaxFunctionEvaluations', 10000, ... % Max function evaluations
%       'PlotFcn', {@psplotbestf, @psplotfuncount}); % Optional: Plot progress

tic;
s_opt = fmincon(@(s) Hybrid_Cost_Function(s), s0, [], [], [], [], [], [], @(s) Hyb
toc;
%s_opt = patternsearch(@(s) Hybrid_Cost_Function(s), s0, [], [], [], [], [], [], @(

%% Simulations
theta0 = s_opt(1);
thetaf = s_opt(2);
lam_0 = [s_opt(3); s_opt(4); s_opt(5)];
t_esc = s_opt(9);
t_capt = s_opt(10);
t_coast = s_opt(11);
tf = t_esc+t_coast+t_capt;

%% ODE 2: Capture + partial coast (t = 0 (LLO) till t = t_capt + t_coast2) (Backwa
m_llo = m_leo - mdot*(t_esc+t_coast+t_capt);
```

```
y0_2 = [x0_2; thetaf; lam_f; 0]; % intial 8 by 1 state

tspan2 = t0:100:tmax;

options_ode1 = odeset('Events', @soi_event, 'RelTol', 1e-6, 'AbsTol', 1e-8); % Eve
[t2, y2] = ode45(@(t,y) Hybrid_ODE_2(t, y, t_capt, m_llo), tspan2, y0_2, options_od

t_coast2 = t2(end) - t_capt;

%% ODE 1: Escape + partial coast (t = 0 (LEO) till t = t_esc + t_coast1) (Forward
y0_1 = [x0_1; theta0; lam_0; 0]; % intial 8 by 1 state
t_coast1 = t_coast - t_coast2;
t1 = t_esc + t_coast1;

tspan1 = t0:100:t1;

%options_ode2 = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[t1_vec, y1] = ode45(@(t,y) Hybrid_ODE_1(t, y, t_esc), tspan1, y0_1);

%% Plot Trajectory and Optimal Thrust angle Profile
r1 = y1(:,1);
vr1 = y1(:,2);

vtheta1 = y1(:,3);
theta1 = y1(:,4);

r2 = y2(:,1);
```

```
vr2 = y2(:,2);
vtheta2 = y2(:,3);
theta2 = y2(:,4);

%[rm_1, thetam_1, vrm_1, vthetam_1] = ECRF_to_MCRF(r1, theta1, v_r1, v_theta1, D, c

% Position in Cartesian Coordinates
x_vec1 = r1.*cos(theta1);
y_vec1 = r1.*sin(theta1);

x_vec2 = r2.*cos(theta2) - D;
y_vec2 = r2.*sin(theta2);

%% for sphere of influence
% Create angle values from 0 to 2*pi
theta_soi = linspace(0, 2*pi, 100);

% Compute x and y coordinates of the circle
x_soi = soi_m*cos(theta_soi)-D;
y_soi = soi_m*sin(theta_soi);

figure;
plot(x_vec1(t1_vec<=t_esc)/Re,y_vec1(t1_vec<=t_esc)/Re);
hold on;
plot(x_vec1(t1_vec(1:end-200)>t_esc)/Re,y_vec1(t1_vec(1:end-200)>t_esc)/Re, 'k--');
plot(x_vec2(t2>t_capt)/Re,y_vec2(t2>t_capt)/Re, 'k--');
plot(x_vec2(t2<=t_capt)/Re,y_vec2(t2<=t_capt)/Re);
plot(x_soi/Re, y_soi/Re, '--')
axis('equal')
xlabel(' X (normalised with Earth radii)')
ylabel(' Y (normalised with Earth radii)')
legend('Thrust arc', 'Coast arc')
title('Optimal retrograde trajectory: rotating coordinates')
xlim([-75, 10])

%% Flight path angle and Thrust (Escape)
lam_vr1 = y1(:,6);
lam_vtheta2 = y1(:,7);

thrust_idx1 = t1_vec <= t_esc;

u1_esc = atan2(-lam_vr1(thrust_idx1),-lam_vtheta2(thrust_idx1)); % Optimal control
u_coast1 = zeros(sum(~thrust_idx1), 1);
u1 = [u1_esc; u_coast1];

gam1 = atan2(y1(:,2), y1(:,3)); % Flight path angle

figure;
plot(t1_vec/t_c, u1*180/pi)
xlabel('time (in days)')
ylabel('Steering and Flight path angles (in deg.)')
hold on
plot(t1_vec/t_c, gam1*180/pi, '--')
legend('Optimal Thrust Angle', 'Flight Path Angle')
```

```
title('Optimal thrust direction and flight path angles: Earth Escape Phase.')
xlim([0, 3])

%% Flight path angle and thrust (Capture)
lam_vr2 = y2(:,6);
lam_vtheta2 = y2(:,7);

thrust_idx2 = t2 <= t_capt;

u1_capt = atan2(-lam_vr2(thrust_idx2),-lam_vtheta2(thrust_idx2))+pi; % Optimal con
u_coast2 = zeros(sum(~thrust_idx2), 1);
u2 = [u1_capt; u_coast2];

gam2 = atan2(y2(:,2), y2(:,3)); % Flight path angle

figure;
plot((tf-t2)/t_c +2, u2*180/pi)
xlabel('time (in days)')
ylabel('Steering and Flight path angles (in deg.)')
hold on
plot((tf-t2)/t_c +2, gam2*180/pi, '--')
legend('Optimal Thrust Angle', 'Flight Path Angle')
title('Optimal thrust direction and flight path angles: Earth Escape Phase.')
%xlim([0, 3])

function dydt = Hybrid_ODE_1(t, y, t_esc)
```

```
% constant values for Earth-Moon system
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system

% Thrusting only upto escape phase
if t<= t_esc
    T = 2942; % Constant Thrust magnitude (in N)
    u = atan2(-y(6), -y(7)); % Optimal control for escape phase
else
    T = 0; % coast phase
    u = 0; % coast phase
end

m_leo = 1e5; % initial mass at LEO (in kg)
mdot = 107.5/3600; % time rate of change of mass due to thrust (in kg/s)
m = m_leo - mdot*t;
a_T = T/m;

% CR3BP EoM in Earth-centered polar rotating frame
dydt = zeros(8,1);

r_msc = (y(1)^2 + 2*D*y(1)*cos(y(4)) + D^2)^(1/2); % r_moon-s/c

% EoM
```

```
dydt(1) = y(2);
dydt(2) = y(3)^2/y(1) - mu_e/y(1)^2 - mu_m*(y(1) + D*cos(y(4)))/r_msc^3 + mu_m*cos(
dydt(3) = mu_m*D*sin(y(4))/r_msc^3 - mu_m*sin(y(4))/D^2 + a_T*cos(u) - 2*omg*y(2) -
dydt(4) = y(3)/y(1);

% Costate
dydt(5) = y(6)*(y(3)^2/y(1)^2 + mu_m/r_msc^3 - 3*mu_m*(y(1) + D*cos(y(4)))^2/r_msc^
dydt(6) = -y(5) + y(7)*(y(3)/y(1) + 2*omg);
dydt(7) = y(6)*(-2*omg - 2*y(3)/y(1)) + y(7)*y(2)/y(1) - y(8)/y(1);
dydt(8) = y(6)*(-mu_m*D*sin(y(4))/r_msc^3 + 3*mu_m*(y(1) + D*cos(y(4)))*D*y(1)*sin(


function dydt = Hybrid_ODE_2(t, y, t_capt, m_llo)

% constant values for Earth-Moon system
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system

% Thrusting only upto escape phase
if t<= t_capt
    T = 2942; % Constant Thrust magnitude (in N)
    u = atan2(-y(6), -y(7))+pi; % Optimal control for escape phase
else
    T = 0; % coast phase
    u = 0; % coast phase
end

mdot = -107.5/3600; % time rate of change of mass due to thrust (in kg/s)
m = m_llo - mdot*t;

a_T = T/m;

% CR3BP EoM in Earth-centered polar rotating frame
dydt = zeros(8,1);

r_esc = (y(1)^2 - 2*D*y(1)*cos(y(4)) + D^2)^(1/2); % r_earth-s/c

% EoM
dydt(1) = -y(2);
dydt(2) = -(y(3)^2/y(1) - mu_m/y(1)^2 - mu_e*(y(1) - D*cos(y(4)))/r_esc^3 - mu_e*co
dydt(3) = -(-mu_e*D*sin(y(4))/r_esc^3 + mu_e*sin(y(4))/D^2 + a_T*cos(u) - 2*omg*y(2
dydt(4) = -y(3)/y(1);

% Costate
dydt(5) = -(y(6)*(y(3)^2/y(1)^2 + mu_e/r_esc^3 - 3*mu_e*(y(1) - D*cos(y(4)))^2/r_es
dydt(6) = -(-y(5) + y(7)*(y(3)/y(1) + 2*omg));
dydt(7) = -(y(6)*(-2*omg - 2*y(3)/y(1)) + y(7)*y(2)/y(1) - y(8)/y(1));
dydt(8) = -(y(6)*(mu_e*D*sin(y(4))/r_esc^3 - 3*mu_e*(y(1) - D*cos(y(4)))*D*y(1)*sin


function J = Hybrid_Cost_Function(s)
```

```
% Extract parameters;
t_esc = s(9);
t_capt = s(10);

% Cost function
J = t_esc + t_capt;



%% Constraint Function (Ensures Boundary Conditions are Satisfied)
function [c, ceq] = Hybrid_Constraint_Function(s, x0_1, x0_2, t0, m_leo, mdot, soi_

% Transform from Earth-Centered to Moon-centered
D = 3.84399*1e8; % Distance between Earth and Moon (in m)
mu_e = 3.986004418*1e14; % in m^3/s^-2
mu_m = 4.9048695*1e12; % in m^3/s^-2
mu = mu_e + mu_m; % in m^3/s^-2
omg = sqrt(mu/D^3); % Constant angular rate of Earth-Moon system

%% Extract parameters
theta0 = s(1);
thetaf = s(2);

lam_0 = [s(3); s(4); s(5)];
lam_f = [s(6); s(7); s(8)];

t_esc = s(9);
```

```
t_capt = s(10);
t_coast = s(11);

%% ODE 2: Capture + partial coast (t = 0 (LLO) till t = t_capt + t_coast2) (Backwa
m_llo = m_leo - mdot*(t_esc+t_coast+t_capt);
y0_2 = [x0_2; thetaf; lam_f; 0]; % intial 8 by 1 state

tspan2 = t0:100:tmax;

options_ode1 = odeset('Events', @soi_event, 'RelTol', 1e-6, 'AbsTol', 1e-8); % Eve
[t2, y2] = ode45(@(t,y) Hybrid_ODE_2(t, y, t_capt, m_llo), tspan2, y0_2, options_od

t_coast2 = t2(end) - t_capt;

%% ODE 1: Escape + partial coast (t = 0 (LEO) till t = t_esc + t_coast1) (Forward
y0_1 = [x0_1; theta0; lam_0; 0]; % intial 8 by 1 state
t_coast1 = t_coast - t_coast2;
t1 = t_esc + t_coast1;

tspan1 = t0:100:t1;

%options_ode2 = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
[~, y1] = ode45(@(t,y) Hybrid_ODE_1(t, y, t_esc), tspan1, y0_1);

%% Extract;
y_final1 = y1(end, :);
r1 = y_final1(1);
```

```
v_r1 = y_final1(2);
v_theta1 = y_final1(3);
theta1 = y_final1(4);

y_final2 = y2(end, :);
r2 = y_final2(1);
theta2 = y_final2(4);
v_r2 = y_final2(2);
v_theta2 = y_final2(3);

% Convert from Earth-centered rotating to Moon-Centered Rotating
[rm_1, thetam_1, vrm_1, vthetam_1] = ECRF_to_MCRF(r1, theta1, v_r1, v_theta1, D, on

%% Constraint
c = []; % No inequality constraints
ceq = [rm_1 - r2; mod(thetam_1, 2*pi) - mod(theta2, 2*pi); vrm_1 - v_r2; vthetam_1

function [value, isterminal, direction] = soi_event(t, x)
    soi_m = 66193923.1913185; % Moon's SOI radius (km)
    r = x(1);  % Radial distance from Moon
    value = r - soi_m;  % Stop when r >= soi_moon
    isterminal = 1;         % Terminate integration
    direction = 1;          % Detect positive crossing (exiting SOI)
end

function [rM, thetaM, vrM, vthetaM] = ECRF_to_MCRF(rE, thetaE, vrE, vthetaE, D, omg
    % Step 1: Convert Earth-centered polar to Cartesian
```

```
xE = rE.*cos(thetaE);
yE = rE.*sin(thetaE);

vxM = vrE .*cos(thetaE) - vthetaE.*sin(thetaE);
vyM = vrE.*sin(thetaE) + vthetaE.*cos(thetaE);

% Step 2: Translate to Moon-centered Cartesian
xM = xE + D;
yM = yE;

% Step 3: Convert Moon-centered Cartesian to polar
rM = sqrt(xM.^2 + yM.^2);
thetaM = atan2(yM, xM);

vrM = vxM.*cos(thetaM) + vyM.*sin(thetaM);
vthetaM = -vxM.*sin(thetaM) + vyM.*cos(thetaM);
```