LEO Constellation End-Of-Life Mission Optimization Tool - Planet Labs Coding Test for Senior Software Engineer, Orbits RD

Dr. Rachit Bhatia

22nd January 2025

Contents

Chapter 1

Introduction

1.1 Motivation

Constellation management, space traffic management, and end-of-life disposal of satellites for ensuring space sustainability are active research areas and are critical for space safety. This project develops a computational framework for planning constellation deorbiting missions, focusing on:

- Safe end-of-life disposal
- Optimized fuel consumption
- Ensuring controlled atmospheric reentry

1.2 Objectives

The primary objectives of this simulation tool are:

- Develop a reusable framework for end-of-life mission planning of LEO satellites
- Enable controlled deorbit while maximizing fuel consumption for safe disposal
- Use simplified dynamics with advanced orbital mechanics and numerical integration
- Calculate optimal burn sequences to lower satellite orbits and manage propellant

1.3 Mission

This tool provides a comprehensive simulation for planning satellite deorbiting missions. Key capabilities include:

- Compute optimal burn start times to achieve target altitudes
- Model atmospheric drag and thrust effects on orbital dynamics
- Manage propellant consumption during the deorbiting process
- Ensure controlled and secure end-of-life disposal of LEO satellites

Chapter 2

Orbital Mechanics

2.1 Coordinate Frames

To set up the algorithm classical orbital elements are used because of the ease with which one can derive intuitive sense of orbit from these elements. In the given problem, because of the scale and complexity, simplified dynamics are used. Classical orbital elements and RTN frame are best suitable in this scenario as one can easily decouple major perturbations and it is easy to track the motion of the object over time.

An RTN frame is shown in Figure 2.1

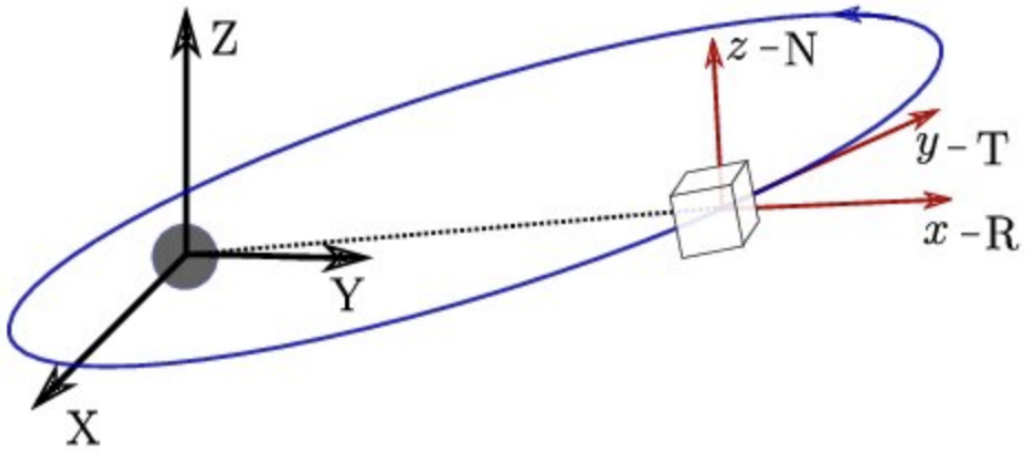

Figure 2.1: Radial-Tangential-Normal (RTN) Coordinate Frame

2.2 Orbital Parameters

Classical orbital elements with following equations are used in the simulation:

$$p = a(1 - e^2) \tag{2.1}$$
$$r = \frac{p}{1 + e\cos(\nu)} \tag{2.2}$$
$$h = \sqrt{\mu p} \tag{2.3}$$
$$c = I_{sp} g_0 \tag{2.4}$$
$$F = ma \tag{2.5}$$

where p is the semi-latus rectum, r is the radius, h is the specific angular momentum, c is the effective exhaust velocity, and F is the thrust force.

2.3 Gauss Planetary Equations

The core of the simulation is based on the Gauss planetary equations:

$$\frac{da}{dt} = \frac{2a^2}{h}\left(e\sin\nu \cdot a_R + \frac{p}{r}a_T\right) \tag{2.6}$$
$$\frac{de}{dt} = \frac{p}{h}\left(\sin\nu \cdot a_R + \frac{\cos\nu + e\cos\nu}{1 + e\cos\nu} \cdot a_T\right) \tag{2.7}$$
$$\frac{di}{dt} = \frac{r}{h}a_N\cos(\theta) \tag{2.8}$$
$$\frac{d\Omega}{dt} = \frac{r}{h}a_N\frac{\sin(\theta)}{\sin(i)} \tag{2.9}$$
$$\frac{d\omega}{dt} = \frac{1}{he}\left(-p\cos\nu a_R + (p + r)\sin\nu a_T\right) - \frac{d\Omega}{dt}\cos(i) \tag{2.10}$$
$$\frac{d\nu}{dt} = \frac{h}{r^2} - \frac{d\omega}{dt} - \frac{d\Omega}{dt}\cos(i) \tag{2.11}$$

where a, e, i, Ω, ω, and ν are the orbital elements, r is the radius, h is the specific angular momentum, and a_R, a_T, and a_N are the components of the acceleration vector in the RTN frame.

Gauss's planetary equations provide an analytical solution to myriad of orbital dynamics problems. However, it is important to remember that this system of equations are based on some key assumptions. These assumptions include that the orbit follows Kepler's laws and can be described by slowly varying osculating elements, with perturbations being relatively weak compared to the central gravitational force. The equations assume that the perturbing forces can be decomposed into radial, transverse, and normal components, and that the perturbations are small enough to allow for linearization. The model also relies on the orbit existing in a two-dimensional plane within three-dimensional space, consistent with the conservation of angular momentum. Additionally, it assumes a specific relationship between position and velocity vectors described by Lagrange coefficients. For initial orbit determination, the equations assume negligible curvature in the observed arc. Using these simplifications we are able to effectively describe orbital element changes over time due to perturbing forces.

2.4 Atmospheric Modeling

The atmospheric density is modeled using an exponential decay function:

$$\rho(h) = \rho_{60} \exp\left(-\frac{h - h_{60}}{H}\right) \tag{2.12}$$

where:

- $\rho_{60} = 3.206 \times 10^{-4}$ kg/m^3 is the density at 60 km altitude
- $H = 7.714 \times 10^3$ m is the scale height
- $h_{60} = 60 \times 10^3$ m is the reference altitude

2.5 Drag Acceleration

The drag acceleration is computed using the formula:

$$a_d = -\frac{1}{2}\frac{\|v\|^2 C_D S}{m}\hat{v} \tag{2.13}$$

where a_d is the drag acceleration, $\|v\|$ is the velocity magnitude, C_D is the drag coefficient, S is the cross-sectional area, and m is the satellite mass.

Chapter 3: Methodology

3.1 Assumptions

To simplify the problem while maintaining accuracy, the following assumptions are made:

- The satellite is modeled as a rigid body with a constant cross-sectional area and mass.
- Atmospheric density follows an exponential decay model and is assumed to be uniform for a given altitude.
- The thrust system operates with a constant specific impulse, ensuring consistent efficiency throughout the deorbit process.
- Perturbations such as solar radiation pressure, J2 effects, and third-body interactions are neglected to focus on atmospheric drag and thrust effects.
- Orbital decay is modeled using classical orbital elements, with changes governed by Gauss's planetary equations.

3.2 Numerical Methods

The simulation employs numerical methods to solve differential equations governing orbital motion:

- **Numerical Integration:** The fourth- and fifth-order Runge-Kutta (RK45) method is used for integrating the equations of motion, ensuring accuracy and stability for varying time steps.
- **Atmospheric Density Model:** The exponential decay function for atmospheric density, $\rho(h) = \rho_{60} \exp\left(\frac{-h-h_{60}}{H}\right)$, is implemented to calculate drag effects based on altitude.
- **Drag Acceleration:** The drag force is computed iteratively using $a_d = -\frac{1}{2}\|v\|^2 \frac{C_D S}{m} \hat{v}$, where velocity magnitude, drag coefficient, satellite cross-sectional area, and mass are inputs.
- **Thrust Modeling:** The specific impulse (I_{sp}) and propellant mass are used to calculate thrust and its influence on the satellite's trajectory.

3.3 Optimization Algorithm

The mission optimization framework involves the following steps:

1. **Initialization:** Define satellite parameters, including mass, cross-sectional area, initial orbital elements, and specific impulse.
2. **Solve Initial ODE:** Use RK45 to propagate the satellite's orbit without thrust to determine the baseline trajectory.
3. **Optimize Burn Start Time:** Apply an iterative optimization algorithm (e.g., gradient descent or genetic algorithms) to determine the optimal burn start time that minimizes propellant usage while achieving target orbital decay.
4. **Convergence Check:** Evaluate convergence criteria based on the deviation from target altitude and fuel consumption. If criteria are unmet, update parameters and iterate.
5. **Visualization:** Generate trajectory plots and console outputs for analysis and validation.

The flow of the optimization process is illustrated in Figure 2.2.

Chapter 4: Results and Discussion

4.1 Simulation Outputs

The developed tool provides the following outputs for various test cases:

1. **Optimal Burn Start Times:** The simulation identifies the ideal times to initiate thrust to achieve controlled reentry.

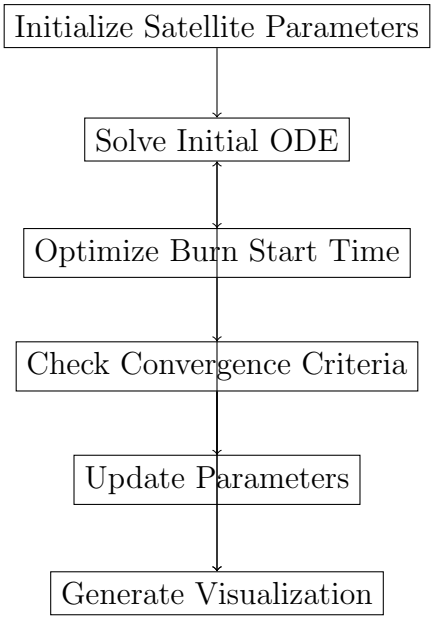

Figure 2.2: Deorbiting Optimization Algorithm Flow

2. **Remaining Propellant:** The tool calculates the amount of propellant consumed and remaining after completing the deorbit maneuver.

3. **Orbital Trajectory Visualization:** Plots of the satellite's orbital trajectory over time, showing decay due to atmospheric drag and thrust.

4.2 Example Case

For an example input, the initial orbital parameters and spacecraft parameters shown in Figure 2.3 are used. The corresponding results of convergence are shown in Figure 2.5 and 2.7.

Figures 2.4 and 2.6 illustrate trajectory evolution from the both precision script and regular script. It is to be noted that precision script avoids approximation of inertial position and velocity and takes into account the rotation of Earth's atmosphere, because of which it's results are more accurate.

4.3 Discussion

The simulation demonstrates the effectiveness of the optimization algorithm in achieving controlled end-of-life disposal:

- The exponential atmospheric density model accurately predicts drag effects at varying altitudes, though deviations are observed at extreme altitudes due to model limitations.

- Propellant consumption is highly sensitive to burn start time, highlighting the importance of precise optimization.

- While the simplified dynamics provide a clear understanding of the deorbit process, incorporating additional perturbations like J2 effects and solar radiation pressure could enhance accuracy.

The results validate the tool's capability to plan end-of-life missions for LEO satellites effectively. Future enhancements to the model and integration with real-time tracking data will improve reliability and usability.

```
main_data = {
  "a": 6578.1363e3,  # Semi-major axis in meters
  "e": 1e-4,  # Eccentricity
  "i": convert_and_range_angle(97.5, 0, np.pi),  # Inclination, degrees
  "RAAN": convert_and_range_angle(100, -np.pi, np.pi),  # RAAN, degrees
  "omega": convert_and_range_angle(200, -np.pi, np.pi),  # Argument of perigee, degrees
  "nu": convert_and_range_angle(50, -np.pi, np.pi),  # True anomaly, degrees
  "S": 2,  # Cross-sectional area in m^2
  "m0": 150,  # Dry mass of the satellite, kg
  "mp": 1,  # Initial propellant mass in kg
  "F": 20e-3,  # Thrust, N
  "Isp": 1000,  # Specific impulse, s
  "Cd": 2.2,  # Drag coefficient
  "Initial_epoch": datetime.strptime("2025-01-01T00:00:00Z", "%Y-%m-%dT%H:%M:%SZ").replace(
      tzinfo=timezone.utc),
  "burn_flag": 0,  # Assuming that no satellite is thrusting initially
  "num_sat": num_sat  # Satellite number
  }
```

Figure 2.3: Example Input

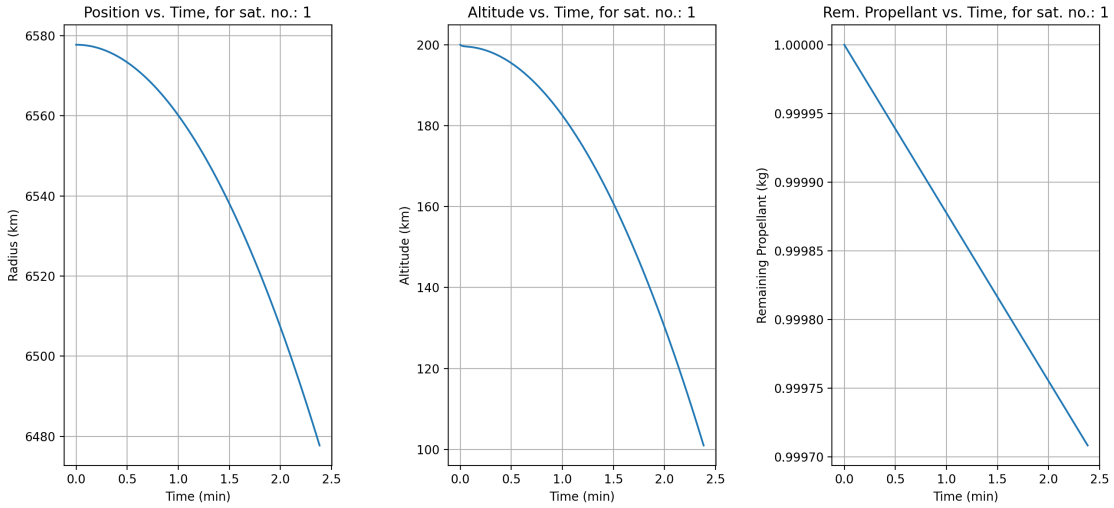

Figure 2.4: Example input results

```
Using 4 threads.
Enter the number of satellites for which deorbiting mission needs to be planned: 1
Use parallel processing? (yes/no): no
Enter details for satellite 1:



Running simulation for satellite 1:



Iteration Number:  1
Iteration Number:  1.
 Reached 100km altitude and have remaining propellant.
 Last altitude: 99.60336679813834 km,
 Optimal burn start time from the initial epoch: 0.0 minutes,
 Remaining propellant: 0.9997064220183486 kg
Optimal burn start time: 0.0 minutes
Remaining propellant: 0.9997064220183486 kg
```

Figure 2.5: Console results for example input results

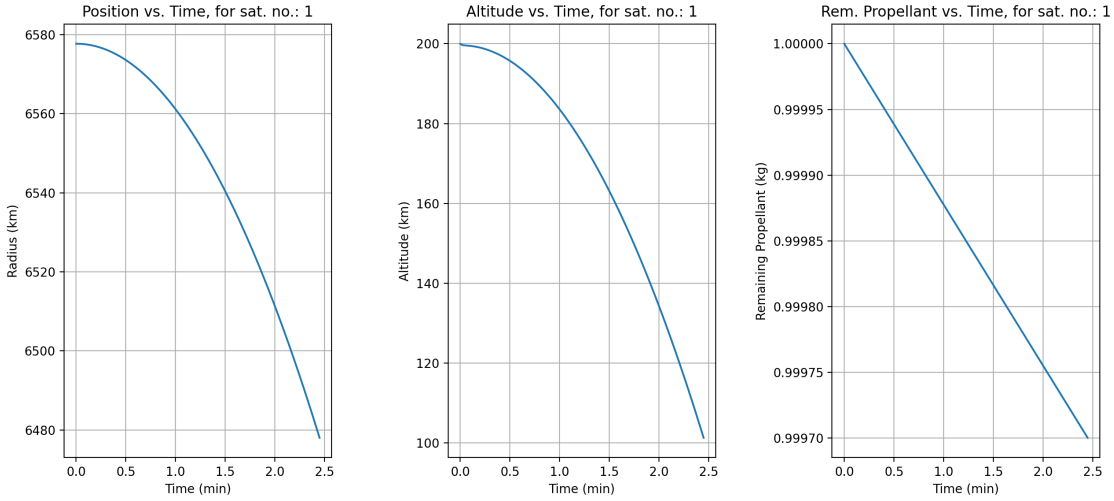

Figure 2.6: Example input results from precision script

```
Using 4 threads.
Enter the number of satellites for which deorbiting mission needs to be planned: 1
Use parallel processing? (yes/no): no



Running simulation for satellite 1:



Iteration Number:  1
Iteration Number:  1.
 Reached 100km altitude and have remaining propellant.
 Last altitude: 99.87369538095885 km,
 Optimal burn start time from the initial epoch: 0.0 minutes,
 Remaining propellant: 0.9996982670744134 kg
Optimal burn start time: 0.0 minutes
Remaining propellant: 0.9996982670744134 kg
```

Figure 2.7: Console results for example input results from precision script