Week 2 Report

Dhanush Balusa

May 27, 2025

Research

Here are some notes/resources to patch up the concept gaps I had from week 1.

TLE:

- `https://web.archive.org/web/20000301052035/http://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html`
- `https://www.mathworks.com/help/satcom/gs/satelliteScenario-key-concepts.html#mw_e08739b4-c5da-4983-898b-56e18cc71f87`

Coordinate System: Geocentric Celestial Reference Frame

- X-axis: points toward the vernal equinox (the intersection of the Earth's equatorial plane and the ecliptic plane at J2000.0).
- Y-axis: lies in the equatorial plane and 90° east of the X-axis.
- Z-axis: points toward the Celestial North Pole (aligned with Earth's mean rotation axis at epoch J2000.0).

SGP4: units: metric

Other resources:

- `https://celestrak.org/columns/v04n03/`

Previous Code

My code from the previous week calculated the approach conditions between the ISS and NOAA 15.

```python
from skyfield.api import EarthSatellite, load
from datetime import datetime
import numpy as np

# TLE (for the ISS)
tle_iss = [
    "ISS (ZARYA)",
    "1 25544U 98067A   24140.51005787  .00004250  00000+0  85977-4 0  9991",
    "2 25544  51.6421 160.9324 0004693 293.4370 181.2067 15.50367430441329"
]

# Load a satellite
sat_iss = EarthSatellite(tle_iss[1], tle_iss[2], tle_iss[0])
ts = load.timescale()
t = ts.utc(2025, 5, 20, 12, 0, 0)  # a UTC time
geo_iss = sat_iss.at(t)

# Get position and velocity
position_km = geo_iss.position.km
velocity_kmps = geo_iss.velocity.km_per_s

print("Position (km):", position_km)
print("Velocity (km/s):", velocity_kmps)

# Load another satellite
tle_noaa15 = [
    "NOAA 15",
    "1 25338U 98030A   24140.39692130  .00000083  00000+0  68134-4 0  9991",
    "2 25338  98.7390 136.7234 0011536 170.2362 189.8990 14.25766605840442"
]

sat_noaa15 = EarthSatellite(tle_noaa15[1], tle_noaa15[2], tle_noaa15[0])
geo_noaa15 = sat_noaa15.at(t)

v1 = np.array(geo_iss.velocity.km_per_s)
v2 = np.array(geo_noaa15.velocity.km_per_s)

relative_velocity = v1 - v2
approach_speed = np.linalg.norm(relative_velocity)

print("Relative position vector (km):", geo_iss.position.km - geo_noaa15.position.km)
print("Relative velocity vector (km/s):", relative_velocity)
print("Approach speed (km/s):", approach_speed)
```

Figure 1: Test code from week 1 calculating the approach conditions between ISS and NOAA15

Code Changes

Based on the feedback from last week's team meeting, I decided to make the following changes:

1. **My first change was to implement the space-track.org API to get TLE data**

   Issue: The EarthSatellite (SGP4) wants 3 arguments but the API call only returns the 2 lines (not the satellite's name).

   Fix: However, I found out a work around be prepending the name:

   ```
   tle_lines = fetch_tle(norad_id, username, password)
   tle_iss = ["ISS (ZARYA)"] + tle_lines
   ```

   Note: I need to check if the TLE data is fetched correctly.

2. **My second change was to loop through a bunch of LEO satellites. I did this with an array of satellites and their catalog number.**

   Note: I need to loop through relevant satellites and potentially go through the whole database.

3. **My third change was to loop through a wide time range.**

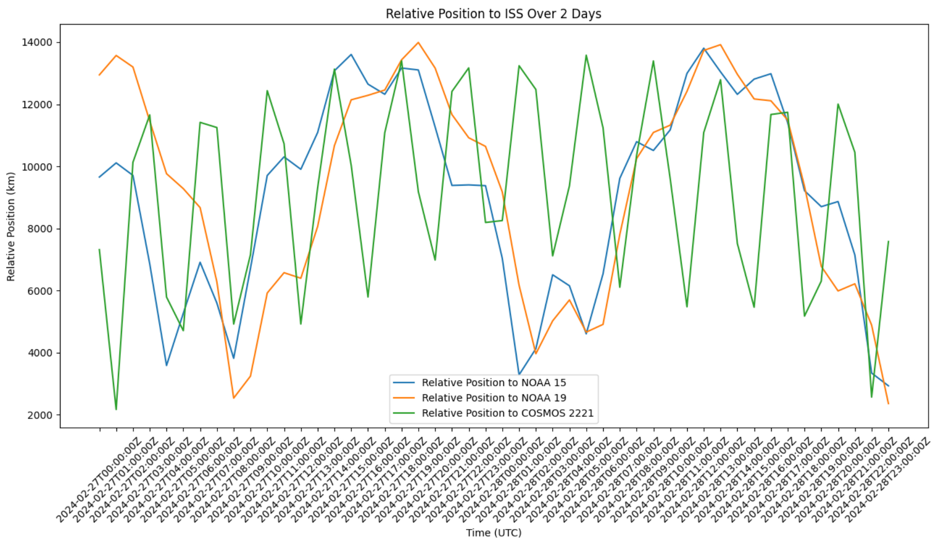

Figure 2: Relative position of ISS and other LEO satellites over time

Issue: When I looped through a huge time range, the plot became very hard to read. However, this also seems to be because I am calculating the approach conditions wrong (more on this in #5), so I need to go back and check if the SGP4 is working.

4. **My fourth step is to graph the relative position vs time to find close approaches**

   Issue: VS Code has a problem compiling the code but using Mac's Terminal worked fine. I need to check which change I made in this step is causing VS Code to not compile.

   Improve: Understand why the relative position is not periodic. Is it because of drag? Or the orbits are different speeds so can't really be a periodic motion when relative to each other?

5. **My fifth step is to check a previous collision in history to cross check if my code works**

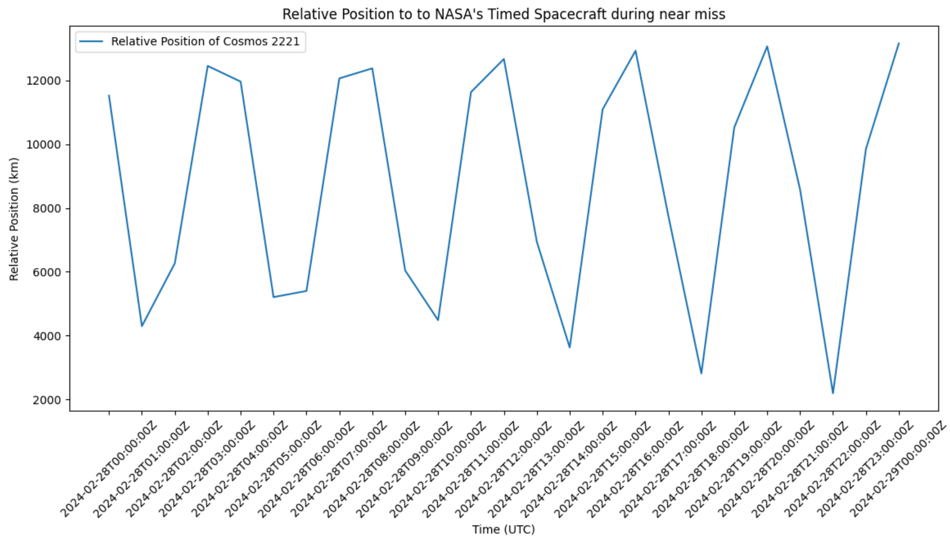

Figure 3: Relative position of NASA's Timed satellite and Cosmos 2221 on 2/28/2024

Issue: I tried a real example of a near miss with NASA's Timed satellite and Cosmos 2221 (with a <30m distance), but the graphs don't look accurate. It is weird that the relative position can change 10,000km within hours. The calculations don't look right.

Improve: I need to cross-check the orbits with GMAT to see the near miss at 6:30 UTC on 2/28/2024. I also need to check if my SGP4 calculations are accurate. Also, could this be a problem with the SGP4 model's error? Another thing to consider is that TLE's are updating frequently, the TLE I used is from 5/28/2025.

Next Steps

- Figure out what satellites I want to compare with
- Figure out if I am using SGP4 correctly, and that the encountered conditions are correct
- Do I need to access old TLEs to see old near misses/collisions?
- Cross-check with GMAT
- Make my code a usable function so Catherine can just call it. Input? $\rightarrow$ Output (angle and approach speed)
- Research about the accuracies of SGP4 and what it actually does