Week 10 Report

Dhanush Balusa

July 25th, 2025

Research

Orbit Covariance:

1. Adapting Covariance Propagation to Account for the Presence of Modeled and Unmodeled Maneuvers: `https://ntrs.nasa.gov/api/citations/20070018023/downloads/20070018023.pdf`

    - Monte Carlo "Prop–Burn–Prop" truth model gives something solid to compare simpler, faster methods against, so they can choose the best approach for orbit prediction and risk analysis.
        - Prop: Generate many slightly different satellite states (position & velocity) using a known initial covariance to represent real-world estimation errors.
        - Burn: Propagate each orbit to the maneuver point, apply a slightly different delta-V (to mimic execution error), then propagate forward again.
        - Prop: Use the final positions and velocities of all the satellites to calculate the "true" covariance, showing how much uncertainty actually grows because of the imperfect maneuver.
    - The paper calls Monte Carlo the "truth model" because it conveys that it is the best available approximation of reality.
    - Three methods that were tested:
        - Method A (burn, no Q): Models the maneuver exactly but ignores any uncertainty. This gives bad results if the maneuver isn't perfect.
        - Method B (no burn, Q): Doesn't model the maneuver but adds uncertainty as noise. Gives better uncertainty but the orbit prediction is off.
        - Method C (burn, Q): Models the maneuver and adds some noise to account for execution errors. Matches both the predicted orbit and the uncertainty well.
    - Q refers to the process noise matrix — it accounts for uncertainties like maneuver execution errors, modeling imperfections, or unmodeled forces.
    - Good covariance estimates are critical for tasks like collision avoidance or tracking satellites after maneuvers.
    - The test used basic physics (no atmospheric drag, etc.), but the idea works with more complex models too.

2. Estimation Strategies for Orbit Determination of Applications Satellites: https://ntrs.nasa.gov/api/citations/19740011388/downloads/19740011388.pdf

    - Covariance matrix was used to evaluate how errors in initial conditions, tracking data, and force models (e.g. gravity) influence orbit accuracy.
    - For satellites like GEOS-C, which operate in long-arc missions (7 days), uncertainties in the Earth's gravity field model are more significant than measurement noise from ground stations.
    - The analysis reveals the radial direction is the most sensitive to geopotential errors, while cross-track and along-track components are less affected.
    - Covariance analysis can be easily added to existing orbit determination software using stored normal and state transition matrices, enabling quick computation of error growth and parameter sensitivity.
    - Once matrices are saved, many estimation strategies can be tested efficiently, making it a low-cost way to optimize orbit determination accuracy. However, an analyical solution is a very complex process.

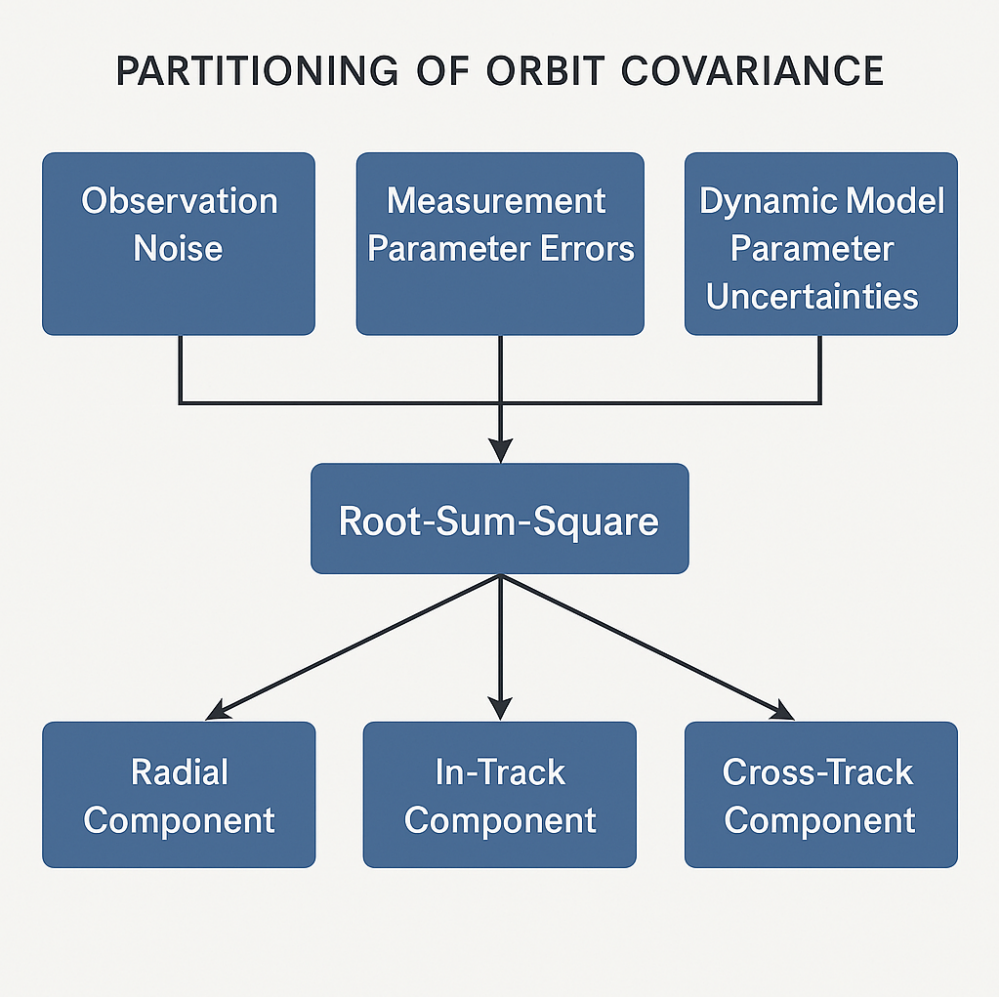

Figure 1: Simplified ChatGPT generated diagram to explain the breakdown of different types of covariance and how it's generated.

Extra resources of analytical covariance:

- Informing Spacecraft Maneuver Decisions to Reduce Probability Informing Spacecraft Maneuver Decisions to Reduce Probability of Collision: `https://scholar.afit.edu/cgi/viewcontent.cgi?params=/context/etd/article/1742`
- A Method for Calculating Collision Probability Between Space Objects: `https://arxiv.org/pdf/1311.7216`

How can I propogate covariance?

- Instead of analytic integration (very complicated), using Monte Carlo is the best way.
- Monte Carlo modeling can be used to sample position errors from covariances and count fraction of samples that fall within the collision volume.
- Skyfield doesn't support custom forces or high-precision integration — limiting my ability to account for uncertainties.

- Skyfield's EarthSatellite class works only from TLEs and uses the SGP4 model. Can't directly propagate from arbitrary position-velocity states using EarthSatellite.
- Tools that might be able to do the job:
  - OREKIT (Java or Python wrapper): supports state vector propagation and covariance. Sampling and statistical analysis aspects would be handled using standard Python libraries like NumPy and SciPy. The `PropagatorsParallelizer` can also be leveraged to efficiently run multiple propagations in parallel.
  - Poliastro (Python): allows propagation from state vectors, and can integrate a Monte Carlo loop easily (random sampling and statistical analysis are handled by other Python libraries like NumPy, Matplotlib, or Plotly).

How to identify collision when there is covariance information?

1. Monte Carlo Method for Collision Probability Calculations using 3D Satellite Models: `https://amostech.com/TechnicalPapers/2010/Integrating_Diverse_Data/deVries.pdf`
   - Uses full 6x6 covariance (position & velocity) for each object to sample many possible states using Gaussian noise.
   - Propogates each sample pair to closest approach and checks if any are within the combined collision volume.
   - The fraction of samples that collide gives the PoC (Probability of Collision).
   - Uses 3D models to map important locaations on the satellite (e.g. solar panels, antennas) to determine collision volume.
   - Monte Carlo gives statistical spread, not a single answer, so it can should the likelihood of collision.

Code Changes

I attempted to debug `week6-7_fetch_tle_data.py` to ensure it correctly fetches the TLE sets. Nothing is still working so I will email CelesTrak for help, because the testing API may be different. If all else fails, I will see if they can give access to the main server again.

After some troubleshooting with the new functions and file directory and path the `week6-7_analyze_approa` code is calculating distance, velocity, amd plotting just fine from a file with TLEs that I manually added.

satellites_tle.txt

```
1   COSMOS 2294 (GLONASS)
2   1 22566U 93016A   24170.48645056  .00000045  00000-0  20036-4 0  9991
3   2 22566  64.9811 112.1954 0023207  71.0115 289.4034 13.75554846308110
4
5   COSMOS 2288 (GLONASS)
6   1 22145U 92039A   24170.48664485 -.00000039  00000-0  00000+0 0  9996
7   2 22145  64.9710  41.7854 0026108 156.3245 203.9632  2.13101141155798
```

Figure 2: Example of the TLE file I used to test the code.

```
Enter the NORAD Catalog IDs to analyze (comma-separated): 22566,22145

  Near Miss Detected Between COSMOS 2294 (GLONASS) and COSMOS 2288 (GLONASS):
    - Closest Distance:      18164.217 km
    - Approach Speed:        9.550 km/s
    - Approach Angle:        90.038 degrees
    - Time (UTC):            2025-06-19T13:40:23Z
```

Figure 3: Example of the output from the code.

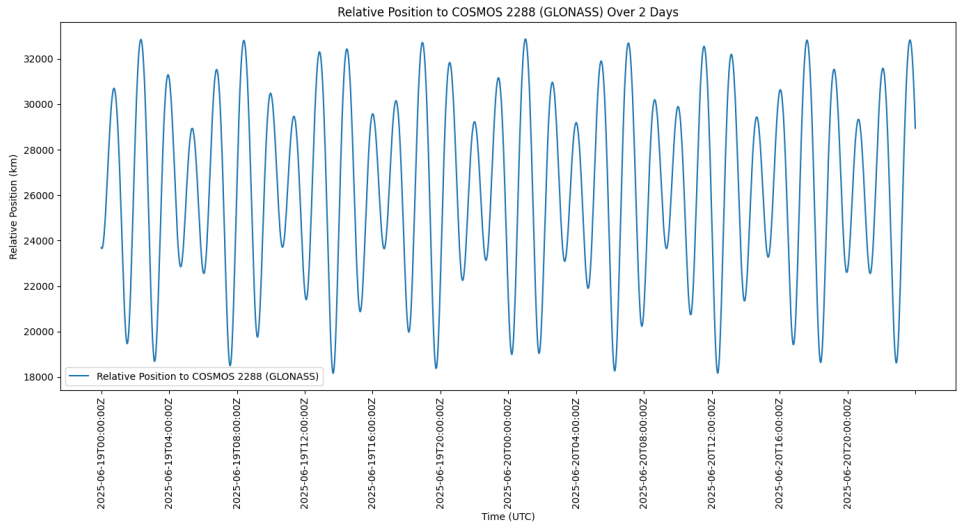

Figure 4: Example of the graph generated by the code.

Next Steps

My Next Steps:

- Fix the TLE fetching code and properly write them into a file.
- Remove the Skyfield dependency from the code and use OREKIT or Poliastro to propagate the TLEs. It will allow me to use covariance and propagate the TLEs from position-velocity states.

Dr. Fan's feedback for next steps:

1. Flowchart of operations (of the code) to better help understand the process.
2. Propose way(s) to fetch TLEs: by class/nature of objects? or by interest or orgin? Time: Jan-1-2025.
3. Find some literature on TLE classes/categories, or how others are fetching TLEs for PoC purposes.
4. TLE Propagation: Also find literature to see how to sample states/TLEs, Gaussian/Random mean and covariance.
5. Propagate IC with initial covariance, for example space objects, and make plots.
6. Start putting together all you have done for the final report.

End goals:

- Loop through all the TLEs in the different categories and calculate the approach conditions for each satellite if it is within a certain distance (e.g. 100 km) of the ISS.
- Incorporate covariance into space object propagation for more realistic collision identification. Also for warning time first propogate for 7 days.
- Make my code a usable function so Catherine can just call it. Some input $\rightarrow$ Output (approach speed and angle).