Satellite Collision Avoidance

Gabriella Armijo

Institute for Computing in Research

August 4, 2022

Outline

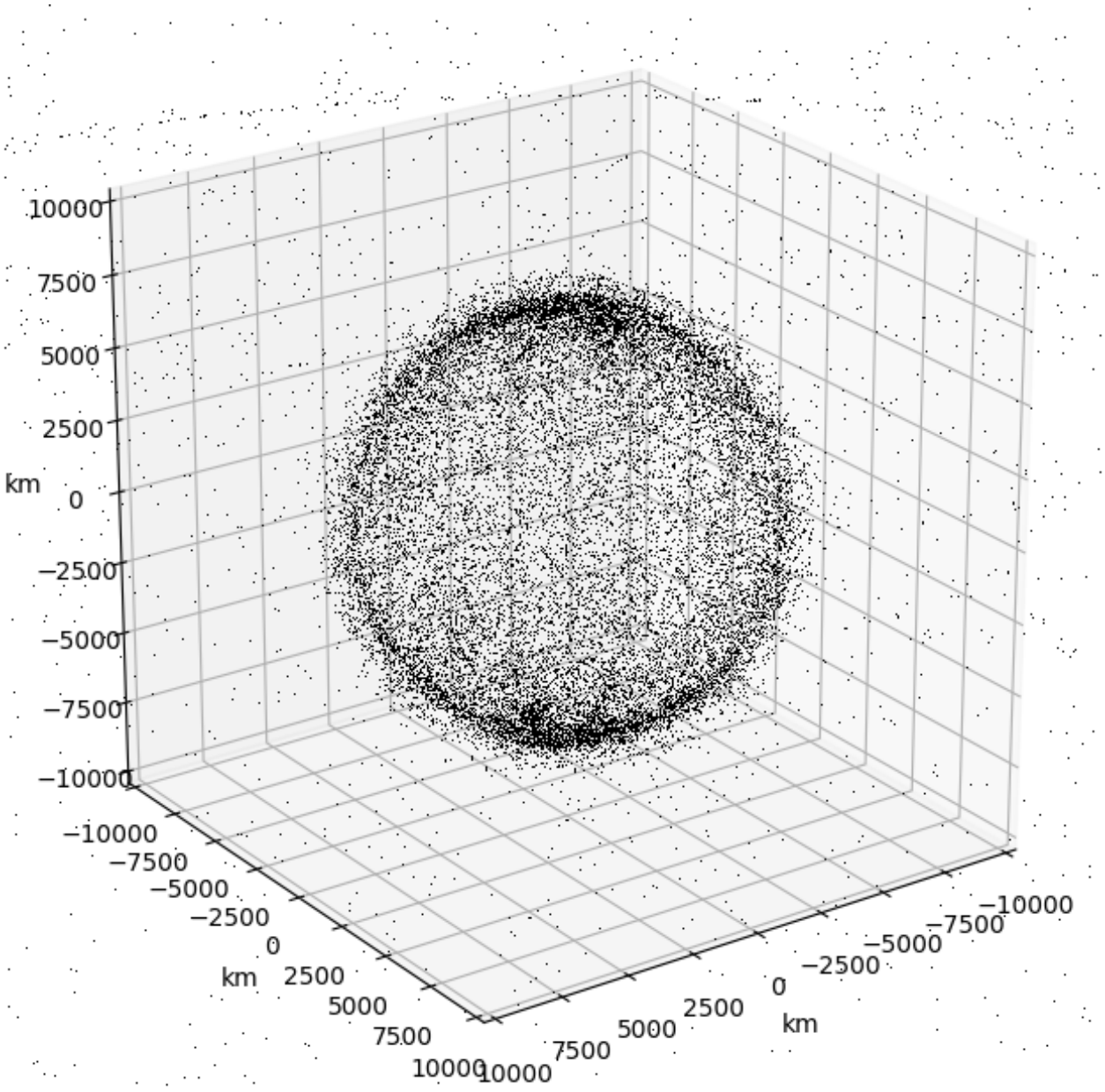
Project Goals

- Track satellites to see how likely they were collide.
- See how often they got within 100 km of each other.
- Analyze those results to see what satellites showed up the most.

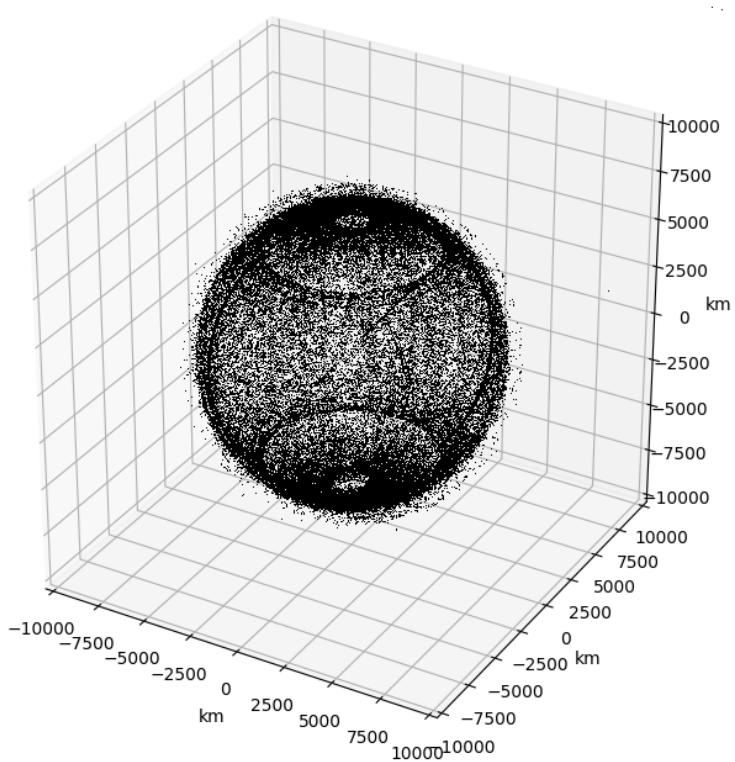
Kessler Syndrome

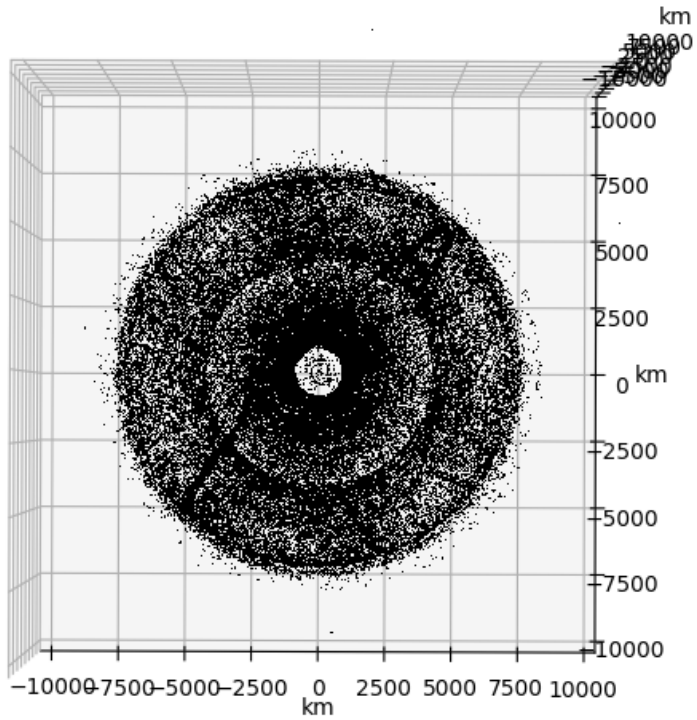
A scenario in which the density of objects in Low Earth Orbit (LEO) is high enough that each collision creates debris that increases the likelihood of more collisions.

3D Plot

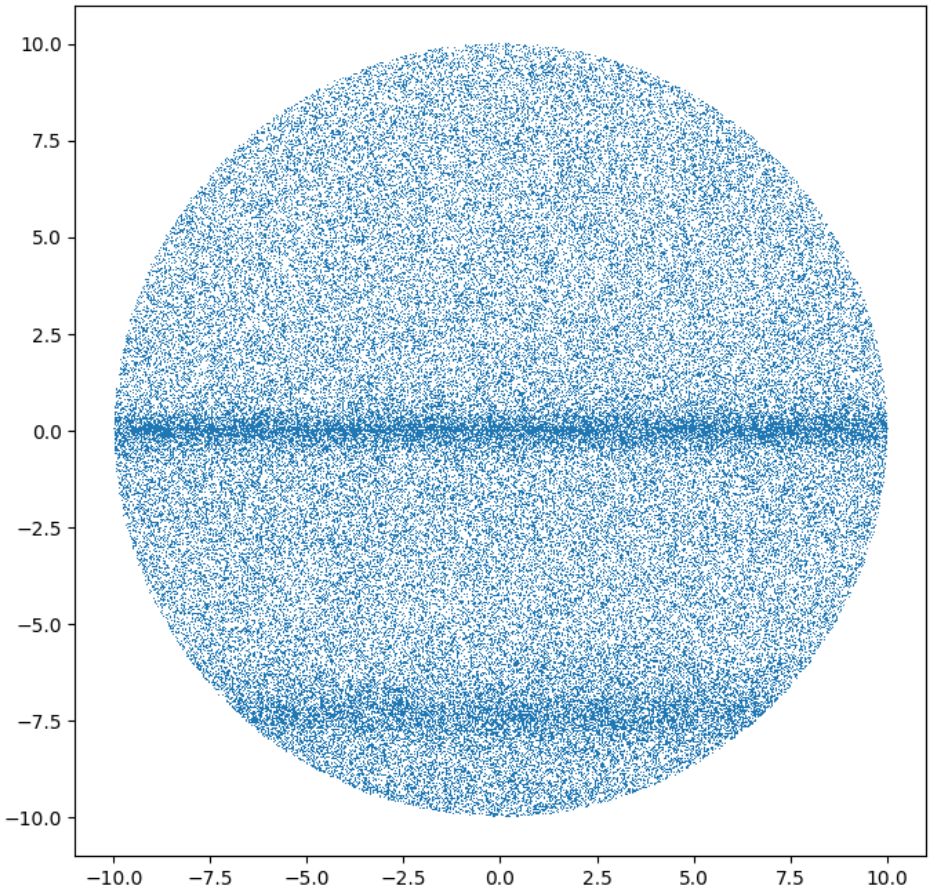

10000
7500
5000
2500
km
0
−2500
−5000
−7500
−10000
−10000
−7500
−5000
−2500
0
km
2500
5000
7500
10000
10000
7500
5000
2500
0
km
−2500
−5000
−7500
−10000

Conjunction Plots





Satellite Point of View



Dot Product


```python
def mindist_and_time(pairpos_vels):
    deltas = pairpos_vels[:, 1] - pairpos_vels[:, 0]
    r = deltas[:, 0, :]
    v = deltas[:, 1, :]
    rnorm = np.sqrt(np.sum(r ** 2, axis=-1))
    vnorm = np.sqrt(np.sum(v ** 2, axis=-1))
    rdotv = np.sum(r * v, axis=-1)
    costheta = rdotv / (rnorm * vnorm)
    sintheta = np.sqrt(1 - costheta ** 2)
    distance = rnorm * sintheta
    # the distance it has to travel to the minimum distance point
    travel = -rnorm * costheta
    time = travel / vnorm
    return distance, time
```

Figure: Dot Product

```python
def pairs_for_time(satellites, time, search_radius=100, maxdistance=10, timestep=10):
    pos_vels = satellitepos_vels(satellites, time)
    if np.any(np.isnan(pos_vels)):
        print(pos_vels.shape, np.count_nonzero(np.isnan(pos_vels[:, 0, 0])), "invalidpositions
        pos_vels = np.nan_to_num(pos_vels)
    pairs, distances, delta_times = findpairs(pos_vels, search_radius=search_radius,
                                              maxdistance=maxdistance, timestep=timestep)
    times = time + delta_times / seconds_per_day
    pair_type = np.dtype([("time_jd", float), ("sat_names", "U30", 2), ("catnums", int, 2),
                          ("positions", float, (2, 3)), ("velocities", float, (2, 3)),
                          ("distance", float), ("extrapolated_distance", float)])
    result = np.zeros(len(distances), dtype=pair_type)
    result["extrapolated_distance"] = distances
    for i in range(len(distances)):
        result[i]["time_jd"] = times[i].tt
        for j, satnum in enumerate(pairs[i]):
            sat = satellites[satnum]
            geocentric = sat.at(times[i])
            result[i]["positions"][j] = geocentric.position.km
            result[i]["velocities"][j] = geocentric.velocity.km_per_s
            result[i]["sat_names"][j] = sat.name
            result[i]["catnums"][j] = sat.model.satnum

    delta_pos = np.diff(result["positions"], axis=1)[:, 0, :]
    new_distance = np.sqrt(np.sum(delta_pos ** 2, axis=-1))
    result["distance"] = new_distance
```

Results


```
[(2459793.49996586, [b'STARLINK-2686', b'STARLINK-4016'], [48464, 52603], [
 (2459793.49995395, [b'COSMOS 2251 DEB', b'FENGYUN 1C DEB'], [36052, 37578],
 (2459793.50000893, [b'STARLINK-1477', b'STARLINK-3763'], [45754, 52556], [
```

```
[[-1794.74737479,  4644.46295733,  4803.81598868], [-1796.996286  ,  4637.41754928,  4798.27287065]], [
], [[-1035.20786393,  1826.2946224 , -6772.34418336], [-1035.9581627 ,  1830.56719824, -6771.60789707]]
[[ 3043.82085744,  3832.24481974,  4890.72523754], [ 3040.55030319,  3828.28038801,  4885.57477159]], [
```

```
-6.96578649,  0.41137309, -2.99202699], [-4.81553841, -5.02496521,  3.04540734]], 9.24237707, 9.24234973),
 [[-4.84987248, -5.65995379, -0.78727613], [ 6.36757837, -3.49630711, -1.91743309]], 4.39999659, 4.40001189)
-6.81385665,  1.79480535,  2.82645312], [-3.28465361,  6.23568948, -2.83285029]], 7.27602526, 7.27602899),
```


```
[(2459793.49996586, [b'STARLINK-2686', b'STARLINK-4016'], [48464, 52603], [
 (2459793.49995395, [b'COSMOS 2251 DEB', b'FENGYUN 1C DEB'], [36052, 37578],
 (2459793.50000893, [b'STARLINK-1477', b'STARLINK-3763'], [45754, 52556], [
```

```
[[-1794.74737479,  4644.46295733,  4803.81598868], [-1796.996286  ,  4637.41754928,  4798.27287065]], [
], [[-1035.20786393,  1826.2946224 , -6772.34418336], [-1035.9581627 ,  1830.56719824, -6771.60789707]]
[[ 3043.82085744,  3832.24481974,  4890.72523754], [ 3040.55030319,  3828.28038801,  4885.57477159]], [
```


```
[-6.96578649,  0.41137309, -2.99202699], [-4.81553841, -5.02496521,  3.04540734]], 9.24237707,
, [[-4.84987248, -5.65995379, -0.78727613], [ 6.36757837, -3.49630711, -1.91743309]], 4.3999965
[-6.81385665,  1.79480535,  2.82645312], [-3.28465361,  6.23568948, -2.83285029]], 7.27602526,
```

Why is this important?

- Prevents Collisions
- Keeping tabs on growing constellations
- Understanding satellite movement

Future Work

- Conjunction Plots
- Starlink orbital readjustments
- Future Collisions

Acknowledgments

I would like to thank my mentor, David Palmer, for everything he has taught me.

I would also like to thank the Institute for Computing in Research and everyone involved for giving me and my fellow interns this opportunity.

References

1 Mann, A., Pultarova, T., Howell, E. (2022, April 14). SpaceX Starlink Internet: Costs, Collision Risks and How it Works. Available at https://www.space.com/spacex-starlink-satellites.html

2 McKnight, D., Shouppe, M., (2021, November 18). Analysis of the Cosmos 1408 Breakup Available at https://leolabs-space.medium.com/analysis-of-the-cosmos-1408-breakup-71b32de5641f

3 Lambert, J. (2018, September). Fengyun-1C Debris Cloud Evolution Over One Decade.

4 Wall, M. (2018, November 15,). Kessler Syndrome and the Space Debris Problem Available at https://www.space.com/kessler-syndrome-space-debris

5 Weeden, B. (2010 November 10,) 2009 Iridium-Cosmos Collision Fact Sheet