Near Earth Comet Simulator Report

Python Decal Spring 2025

Sareena Mann

Github, Demo, Slides

1. Objective

There are about 37,000 active Near Earth Objects, and about 2,500 of which are considered potentially hazardous (CNEOS). Therefore, understanding their orbits and how factors such as eccentricity affect their potential trajectories. In this project report, I outline the approach I took to analyze the trajectories of Near Earth comets. Near Earth objects are defined as objects with trajectories within 1.3 AU of the Sun and therefore within 0.3 AU of the Earth's orbit (UNOOSA). I will use data provided by NASA's Open Portal to animate the orbits of 160 Near Earth Orbits and to simulate how including random noise in the data affects the predicted orbits.

2. Data Used

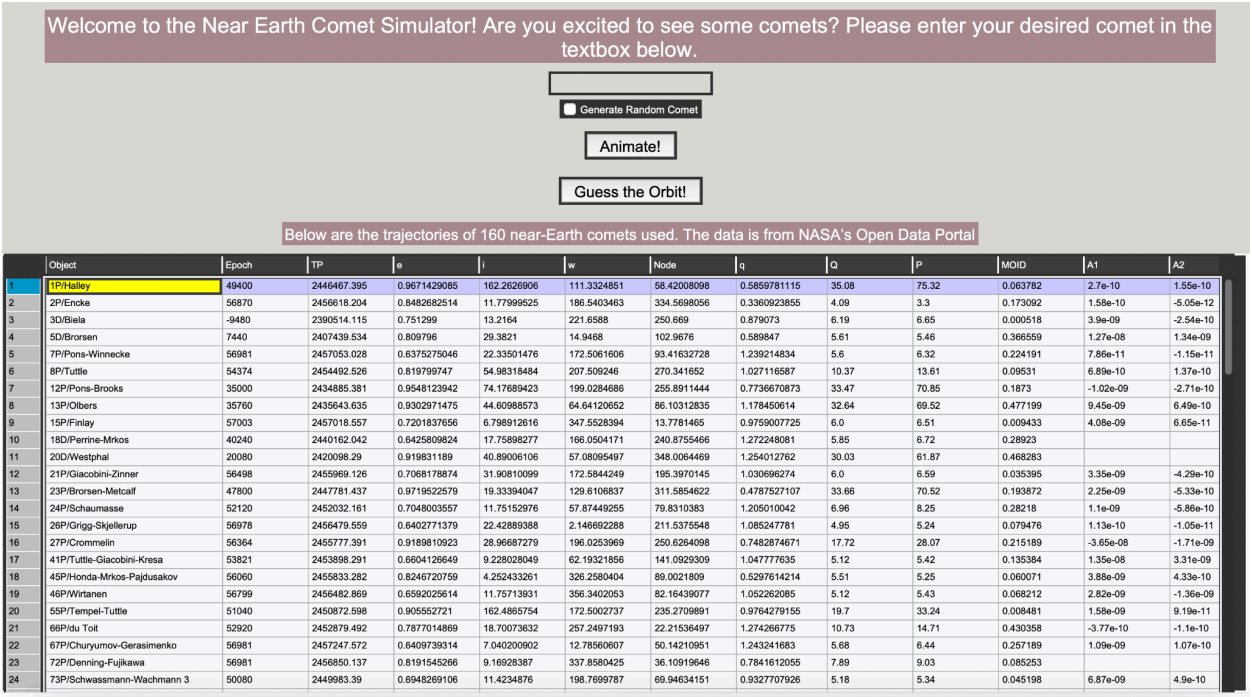
I used data from NASA's Open Portal on Near-Earth Comets. The dataset tracks 15,000 Near-Earth objects, and I extracted 160 comets from the data set to test my algorithm on. For each of the 15,000 object, the following information is provided:

- Epoch: the time in Barycentric Dynamical Time (TDB) when the data was recorded
- **Time of Perihelion passage:** the time when the comet was closest to the Sun in Barycentric Dynamical Time (TDB)
- **Eccentricity of orbit**
- **Inclination of orbit** with respect to the ecliptic plane in degrees
- **Argument of perihelion** (within in the orbital plane) in degrees
- **Node in degrees**
- **The comet's distance at perihelion in AU**
- **The comet's distance at the aphelion in AU**
- **Orbital period in Julian Years**
- Longitude of the ascending node in degrees
- A1: Non-gravitational force parameter A1
- A2: Non-gravitational force parameter A2
- A3: Non-gravitational force parameter A3
- MOID: Minimum orbit intersection distance

In order to clean the data, I first filtered the dataset to only include the 160 comets as those objects were given a name in the dataset. In addition, to animate the orbit of each comet, I only required the information bolded above. To clean this data, I used the pandas library.

3. GUI

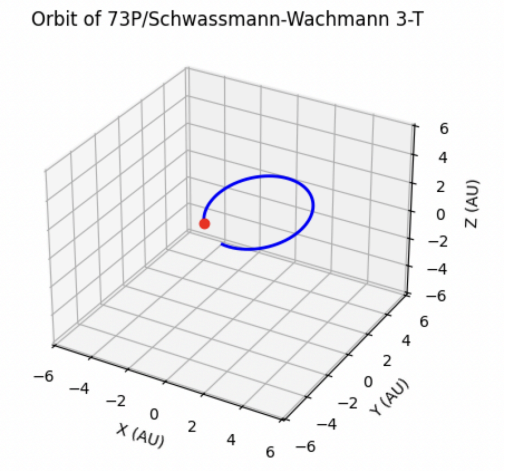
A goal of this project was to create an application where the user could pick or defer to a random comet generator to display a comet. Therefore, I leveraged the Tkinter library, or a gui python library, to render buttons and explain to the user how the application works.



	Object	Epoch	TP	e	i	w	Node	q	Q	P	MOID	A1	A2
1	1P/Halley	49400	2446467.395	0.9671429085	162.2626906	111.3324851	58.42008098	0.5859781115	35.08	75.32	0.063782	2.7e-10	1.55e-10
2	2P/Encke	56870	2456618.204	0.8482682514	11.77999525	186.5403463	334.5698056	0.3360923855	4.09	3.3	0.173092	1.58e-10	-5.05e-12
3	3D/Biela	-9480	2390514.115	0.751299	13.2164	221.6588	250.669	0.879073	6.19	6.65	0.000518	3.9e-09	-2.54e-10
4	5D/Brorsen	7440	2407439.534	0.809796	29.3821	14.9468	102.9676	0.589847	5.61	5.46	0.366559	1.27e-08	1.34e-09
5	7P/Pons-Winnecke	56981	2457053.028	0.6375275046	22.33501476	172.5061606	93.41632728	1.239214834	5.6	6.32	0.224191	7.86e-11	-1.15e-11
6	8P/Tuttle	54374	2454492.526	0.819799747	54.98318484	207.509246	270.341652	1.027116587	10.37	13.61	0.09531	6.89e-10	1.37e-10
7	12P/Pons-Brooks	35000	2434885.381	0.9548123942	74.17689423	199.0284686	255.8911444	0.7736670873	33.47	70.85	0.1873	-1.02e-09	-2.71e-10
8	13P/Olbers	35760	2435643.635	0.9302971475	44.60988573	64.64120652	86.10312835	1.178450614	32.64	69.52	0.477199	9.45e-09	6.49e-10
9	15P/Finlay	57003	2457018.557	0.7201837656	6.798912616	347.5528394	13.7781465	0.9759007725	6.0	6.51	0.009433	4.08e-09	6.65e-11
10	18D/Perrine-Mrkos	40240	2440162.042	0.6425809824	17.75898277	166.0504171	240.8755466	1.272248081	5.85	6.72	0.28923		
11	20D/Westphal	20080	2420098.29	0.919831189	40.89006106	57.08095497	348.0064469	1.254012762	30.03	61.87	0.468283		
12	21P/Giacobini-Zinner	56498	2455969.126	0.7068178874	31.90810099	172.5844249	195.3970145	1.030696274	6.0	6.59	0.035395	3.35e-09	-4.29e-10
13	23P/Brorsen-Metcalf	47800	2447781.437	0.9719522579	19.33394047	129.6106837	311.5854622	0.4787527107	33.66	70.52	0.193872	2.25e-09	-5.33e-10
14	24P/Schaumasse	52120	2452032.161	0.7048003557	11.75152976	57.87449255	79.8310383	1.205010042	6.96	8.25	0.28218	1.1e-09	-5.86e-10
15	26P/Grigg-Skjellerup	56978	2456479.559	0.6402771379	22.42889388	2.146692288	211.5375548	1.085247781	4.95	5.24	0.079476	1.13e-10	-1.05e-11
16	27P/Crommelin	56364	2455777.391	0.9189810923	28.96687279	196.0253969	250.6264098	0.7482874671	17.72	28.07	0.215189	-3.65e-08	-1.71e-09
17	41P/Tuttle-Giacobini-Kresa	53821	2453898.291	0.6604126649	9.228028049	62.19321856	141.0929309	1.047777635	5.12	5.42	0.135384	1.35e-08	3.31e-09
18	45P/Honda-Mrkos-Pajdusakov	56060	2455833.282	0.8246720759	4.252433261	326.2580404	89.0021809	0.5297614214	5.51	5.25	0.060071	3.88e-09	4.33e-10
19	46P/Wirtanen	56799	2456482.869	0.6592025614	11.75713931	356.3402053	82.16439077	1.052262085	5.12	5.43	0.068212	2.82e-09	-1.36e-09
20	55P/Tempel-Tuttle	51040	2450872.598	0.905552721	162.4865754	172.5002737	235.2709891	0.9764279155	19.7	33.24	0.008481	1.58e-09	9.19e-11
21	66P/du Toit	52920	2452879.492	0.7877014869	18.70073632	257.2497193	22.21536497	1.274266775	10.73	14.71	0.430358	-3.77e-10	-1.1e-10
22	67P/Churyumov-Gerasimenko	56981	2457247.572	0.6409739314	7.040200902	12.78560607	50.14210951	1.243241683	5.68	6.44	0.257189	1.09e-09	1.07e-10
23	72P/Denning-Fujikawa	56981	2456850.137	0.8191545266	9.16928387	337.8580425	36.10919646	0.7841612055	7.89	9.03	0.085253		
24	73P/Schwassmann-Wachmann 3	50080	2449983.39	0.6948269106	11.4234876	198.7699787	69.94634151	0.9327707926	5.18	5.34	0.045198	6.87e-09	4.9e-10

Above is the Home Screen of the application. It includes text explaining where to insert the desired comet. There is a textbox where the user can type a comet from the 160 comets listed in the chart at the bottom of the screen. The chart is scrollable and allows the user to see where the animation data is coming from as well as to see how varying variables can affect the orbit. Under the textbook is a "Generate random Comet!" option. If clicked, I use a random seed object to pick a number from 0 to 160 and use the corresponding comet. Finally, the user has the option to "Animate!" or "Guess the Orbit!" The animate feature simply renders the desired comet's orbit in 3D space while the Guess the orbit feature "randomly" distorts the orbital data and compares it to the comet's actual orbit. These features are further explained below.

4. Animation Feature



Output for the orbit of the 73P/Schwassmann-Wachmann 3-T comet

The comet animation feature visualizes the 3D orbital path of a selected near-Earth comet using its orbital parameters from the CSV dataset. By solving Kepler's Equation numerically for a range of mean anomalies, the script compute the comet's position in space throughout its orbit. These positions are then transformed from 2D to 3D space with the consideration of the object's inclination, argument of perihelion, and longitude of the ascending node. These coordinates are calculated with the orbital the orbit_positions function which returns an array of the coordinates.

The animate function then retrieves the comet data, calculates its orbital trajectory, and then uses Matplotlib's FuncAnimation to create a dynamic 3D plot showing the comet tracing its orbit in real time. The head of the comet is highlighted as a red point, and the full path is drawn progressively, providing a clear and engaging visualization of its motion through space.

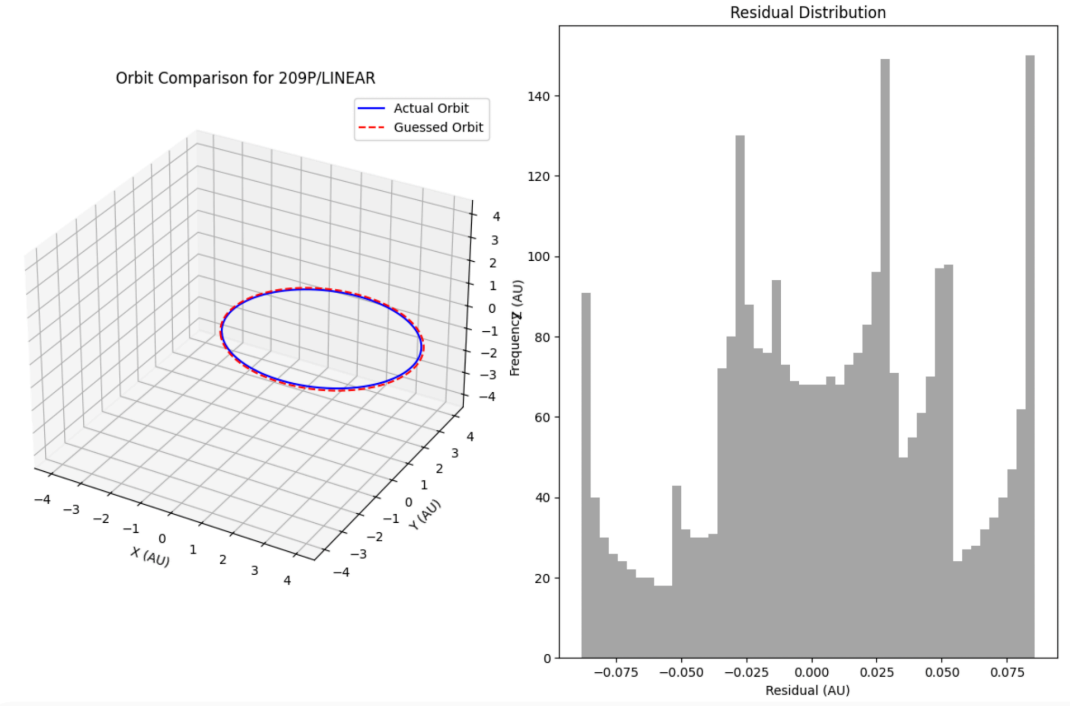
5. Guess Orbit Feature

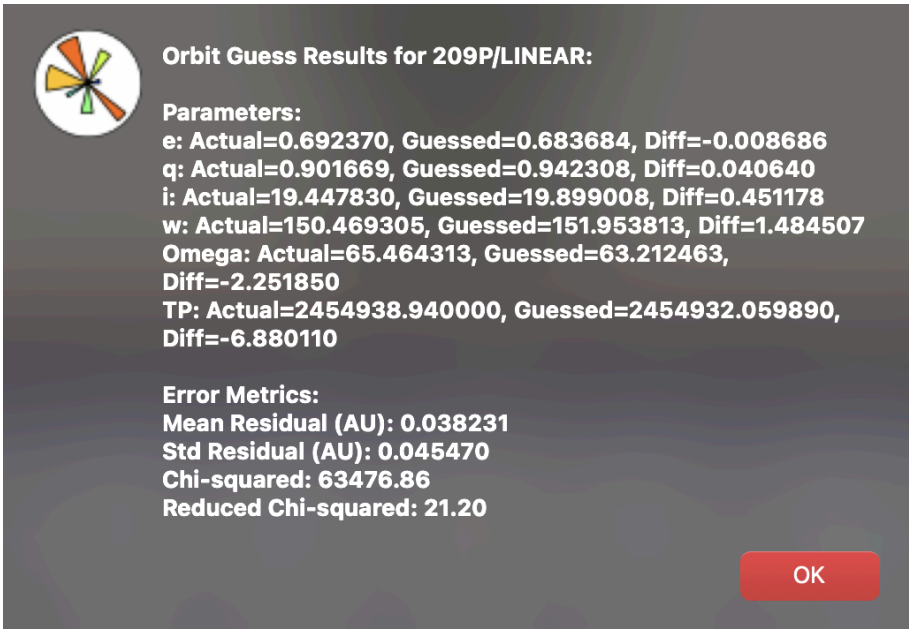
The Guess Orbit feature provides a comprehensive system for analyzing, estimating, and visualizing the 3D orbits of near-Earth comets. It includes four main components:

a. Orbit Estimation with Perturbation (guess_orbit): This function estimates a comet's orbit by altering its actual orbital parameters– eccentricity, perihelion distance, inclination, argument of perihelion, longitude of ascending node, and time of perihelion passage– by a random perturbation factor. The actual and guessed orbits are then calculated using the orbit_positions function (also used by the animation feature). Both parameters sets are then returned along with their 3D position array.

b. Orbit Comparison and Error Analysis (compare_orbits): The function compares the guessed and actual orbits by computing the residuals between the two orbits at each point in time. It also calculates the mean and standard deviation of the

residuals, total chi-squared error, and reduced chi-squared. These metrics provide insight into how accurately the guessed orbit approximates the actual trajectory.





Sample Output for what is rendered by the Visual Comparison component. The top visual includes the guessed v. actual orbit in 3D space as well as the residual plot. The bottom image is message pop up that is rendered which compares the actual and guessed parameters.

c. Visual Comparison (plot_orbit_comparison): This function generates the actual versus guessed comet orbits in 3D space and a histogram of the positional residuals. This helps in visually assessing the quality of the orbital guess and understanding the spatial distribution of errors.
d. Orbit Animation (animate): The animation function reads the comet's orbital data from a CSV file, calculates its 3D orbit, and then animates the comet's motion through space using Matplotlib's FuncAnimation.

6. Summary

Through this project, I was able to apply the data analysis skills learned throughout the semester. In addition, I learned how to use Tkinter and other libraries on the job by reading and looking over a library's spec and functions. I learned how important it is to break up and document one's code into components and to use the idea of abstraction to have the various programs work together.

I successfully designed an application for analyzing and visualizing the orbits of Near Earth comets, using real orbital data and physics-based modeling. Its primary use is educational, exploratory, or research-oriented, allowing users to:

1. Retrieve and visualize comet orbits from a dataset by name, displaying their 3D paths around the Sun using animated plots that would have otherwise taken much longer by hand
2. Estimate of "guess" a comet's orbit by introducing small perturbations to its known orbital parameters. This helps users understand the sensitivity of orbital trajectories to parameter changes.
3. Compare estimated vs. actual orbits through visual plots and statistical analysis, including residual errors and chi-squared metrics, which are useful for evaluating how well an orbital guess fits real data.
4. Explore orbital dynamics in a dynamic, interactive way using Python-based tools (NumPy, Matplotlib, SciPy), making it a valuable application for astronomy students, educators, or enthusiasts seeking insight into celestial mechanics.

Overall, the application serves as a hands-on tool for orbit simulation, parameter sensitivity analysis, and educational visualization of cometary motion in space.

7. Sources

Stephenson, Bruce. *Kepler's Physical Astronomy.* United Kingdom, Princeton University Press, 1994.

Robert.wickramatunga. “United NationsOffice for Outer Space Affairs.” *Near-Earth Objects*, www.unoosa.org/oosa/sk/ourwork/topics/neos/index.html#:~:text=A%20near%2DEarth%20object%20is,kilometres%2C%20of%20the%20Earth’s%20orbit. Accessed 20 Apr. 2025.

“Discovery Statistics.” *NASA*, NASA, cneos.jpl.nasa.gov/stats/totals.html. Accessed 20 Apr. 2025.

“Near-Earth Comets - Orbital Elements - NASA Open Data Portal.” *NASA*, NASA, data.nasa.gov/dataset/near-earth-comets-orbital-elements. Accessed 20 Apr. 2025.