ES98B Group Project – Tutorial

Group SNOE

01

SET-UP

Set-up & installation of the required packages

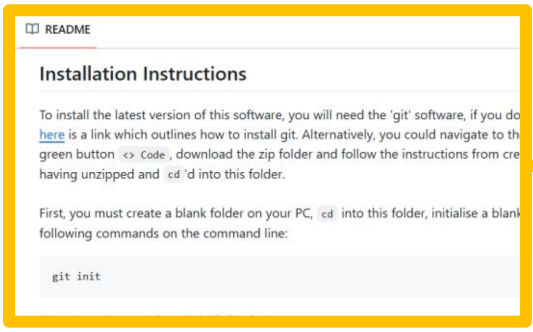
1. To begin, open the GitHub repository page for the project:

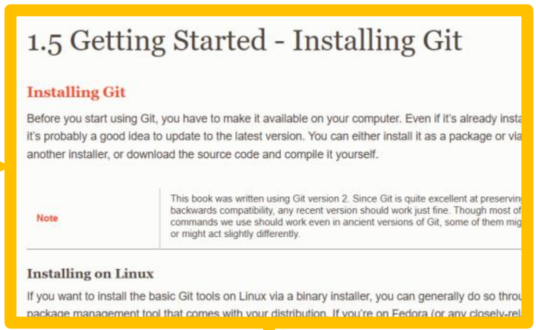
   [De-Orbiting-Satellite-GroupProj: A group project modelling a de-orbiting satellite]

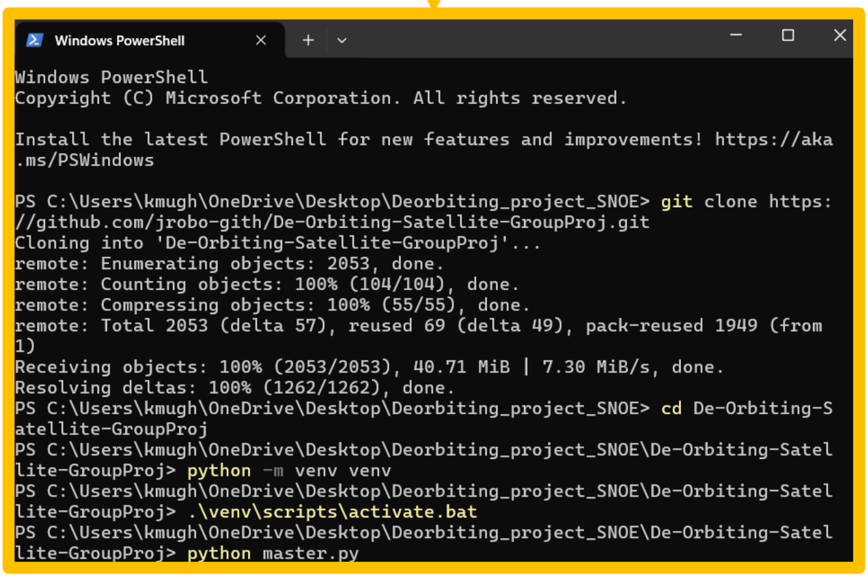
2. Inside the repository, you will find a **README.md** file that contains detailed instructions on how to install the required Python packages.

3. Carefully follow the installation steps listed in the **README**: Create and Activate the virtual environment, then run **Master.py**.

you may refer to the screenshots below to walk you through the process:







```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka
.ms/PSWindows

PS C:\Users\kmugh\OneDrive\Desktop\Deorbiting_project_SNOE> git clone https:
//github.com/jrobo-gith/De-Orbiting-Satellite-GroupProj.git
Cloning into 'De-Orbiting-Satellite-GroupProj'...
remote: Enumerating objects: 2053, done.
remote: Counting objects: 100% (104/104), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 2053 (delta 57), reused 69 (delta 49), pack-reused 1949 (from
1)
Receiving objects: 100% (2053/2053), 40.71 MiB | 7.30 MiB/s, done.
Resolving deltas: 100% (1262/1262), done.
PS C:\Users\kmugh\OneDrive\Desktop\Deorbiting_project_SNOE> cd De-Orbiting-S
atellite-GroupProj
PS C:\Users\kmugh\OneDrive\Desktop\Deorbiting_project_SNOE\De-Orbiting-Satel
lite-GroupProj> python -m venv venv
PS C:\Users\kmugh\OneDrive\Desktop\Deorbiting_project_SNOE\De-Orbiting-Satel
lite-GroupProj> .\venv\scripts\activate.bat
PS C:\Users\kmugh\OneDrive\Desktop\Deorbiting_project_SNOE\De-Orbiting-Satel
lite-GroupProj> python master.py
```

02

Software Overview

Software Main Menu

Once the system is launched, the following main menu will appear:

01

Simulation

Opens the simulation menu, where you will be able select the initial configurations of the simulation run.

02

Instructions

Opens a help window with a simple guide for how to use the system.

03

Credits

Displays the list of contributors to the project.

04

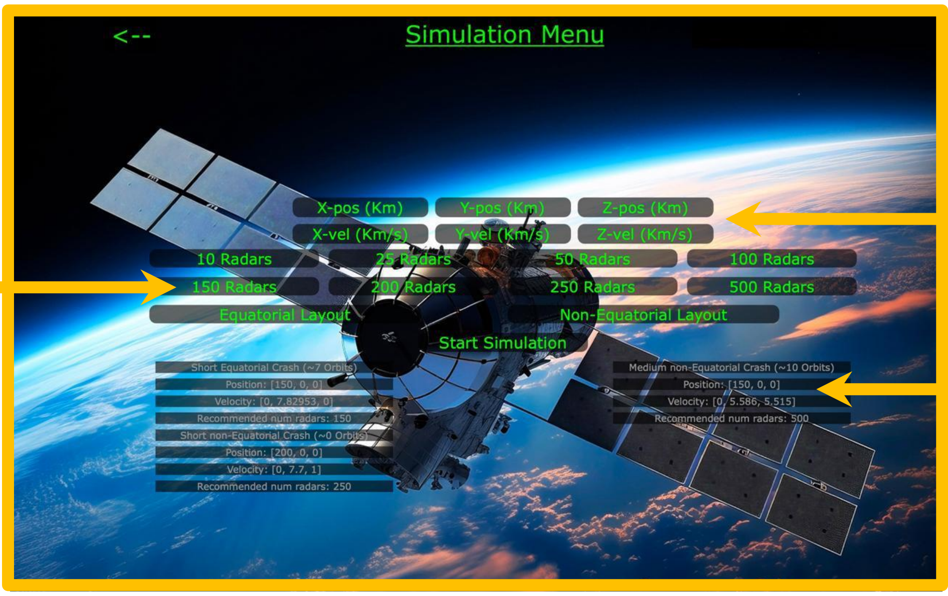
Exit

Closes the application.

Simulation Menu

Once you click “Simulation” from the main menu, you will be taken to the menu shown below, where you can configurate the initial conditions for the satellite and set the numbers of radar stations you want :

1. Enter the satellite’s initial position and velocity. Use the fields labelled X-pos, Y-pos, Z-pos for position. Use X-vel, Y-vel, Z-vel to define the initial velocity vector. These values represent the satellite's state in the ECI coordinate system. You can use the recommended values listed in grey below.



2. Select the number of radar stations and whether they are placed along the equator. This controls how many independent measurements the Kalman Filter receives during the simulation.

   Note: Using more radar stations with non-equatorial distribution generally improves filter accuracy by providing more frequent and diverse measurements.

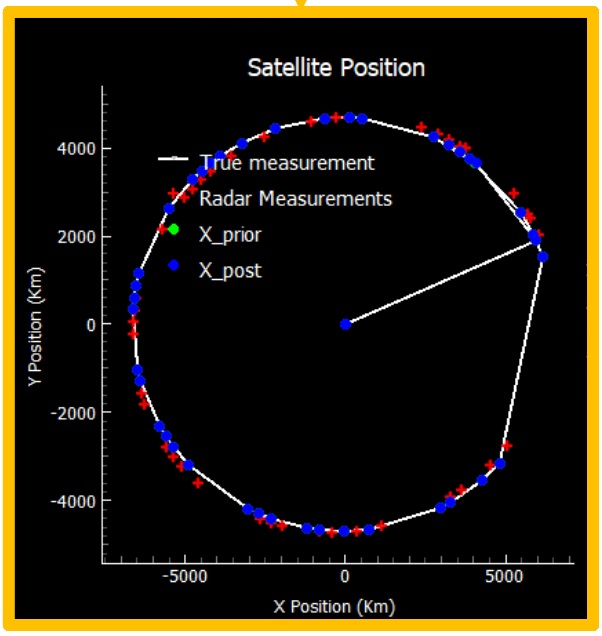
3. Click **“Start Simulation”** to run the software with the specified values.

Simulation – Graph View

When you run the Simulation, you will notice two main options in the top: Graph View and Earth View. Now we present the Graph View:

Graph View displays ten real-time plots that update continuously during execution. These plots provide a live overview of key state variables, estimation errors, and uncertainty metrics, enabling to monitor the filter's performance and system behaviour as the simulation runs.

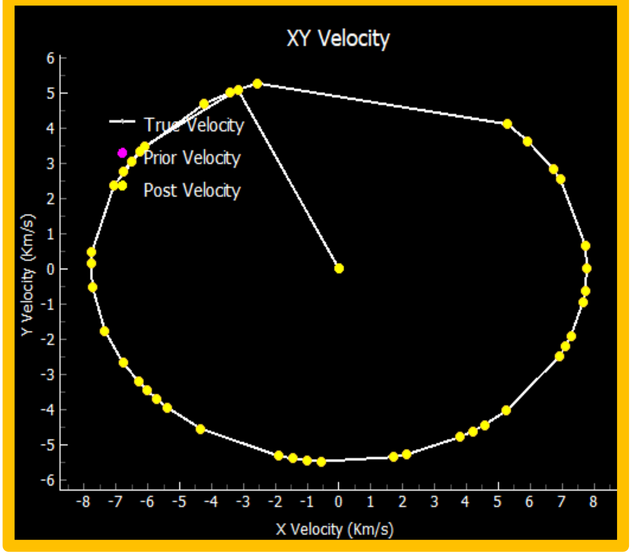
01 Satellite Position Plot



This plot displays the satellite's position in the orbital plane throughout the simulation. The white curve represents the true trajectory generated by the simulation. The red marks are the radar measurements, which are only available when the satellite is within the radar's line of sight. The green markers show the filter's prior predictions, while the **blue** markers show the posterior estimates. When the filter is working well, the predictions converge toward the white trajectory over time, indicating that the UKF is accurately estimating the satellite's state. Gaps in the prediction trace occur in segments where radar measurements are missing, causing the filter to operate in prediction-only mode.

Simulation – Graph View

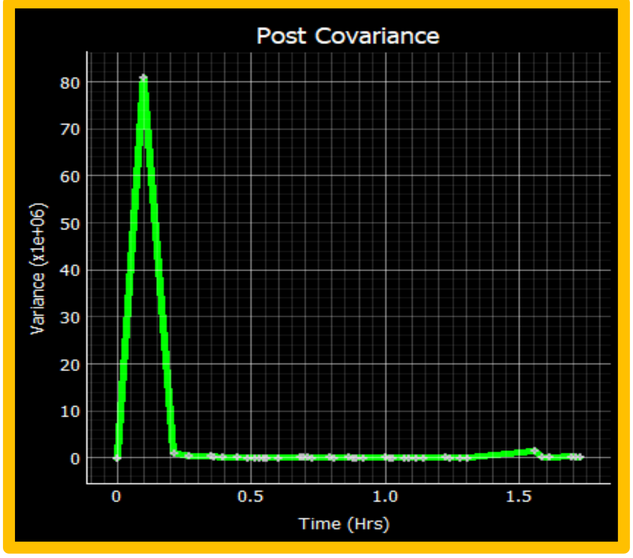
Satellite Velocity Plot 02



This plot visualizes the satellite's velocity in throughout the simulation. The white curve represents the true velocity vector trajectory. The pink markers show the filter's prior velocity estimates, and the yellow markers represent the posterior velocity estimates. A well-performing filter is expected to produce posterior estimates that gradually align with the true velocity.

03 Covariance Trace Plot

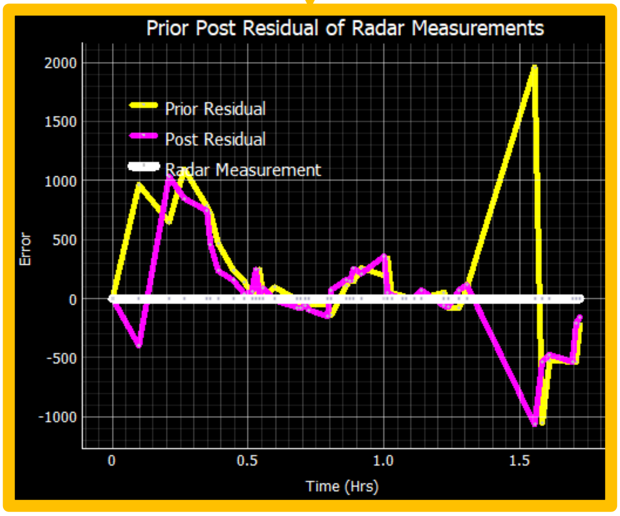
This plot shows the trace of the state covariance matrix **P** over time. The trace represents the total uncertainty across all state dimensions and is used to assess the filter's internal confidence. A decreasing trace indicates that the filter is gradually becoming more certain about its estimates. Sudden rises in the trace may occur during periods without radar updates. Once new measurements are received, the trace decline again as the filter regains information.



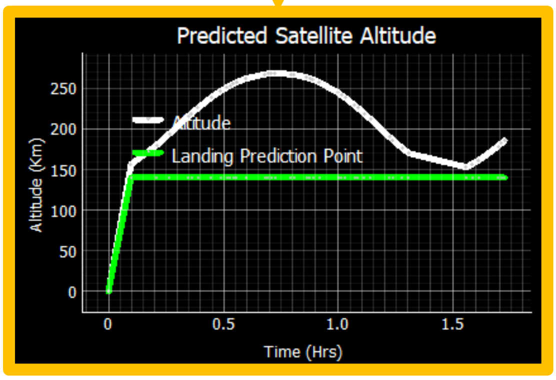
Simulation – Graph View

04 Residual Plot

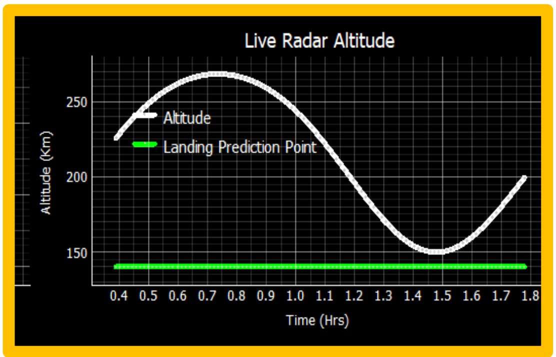
This plot displays the residual error over time between the true state and the filter's estimates. The white line represents the true residual which is zero, the yellow line represents the prior estimate, and the pink line shows the residual from the posterior estimate. As the simulation progresses, we expect these values to approach zero and remain relatively stable.



Estimated Altitude Plot 05



This plot shows the satellite's estimated altitude over time (white curve). The green line marks the 140 km threshold, which triggers the landing prediction process. Once this line is crossed, the system begins continuously estimating the landing site and time.

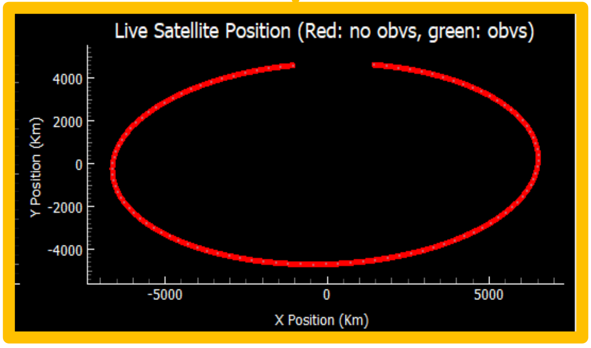


Live Radar Altitude Plot 06

This plot shows the altitude of the satellite as measured by the radar.

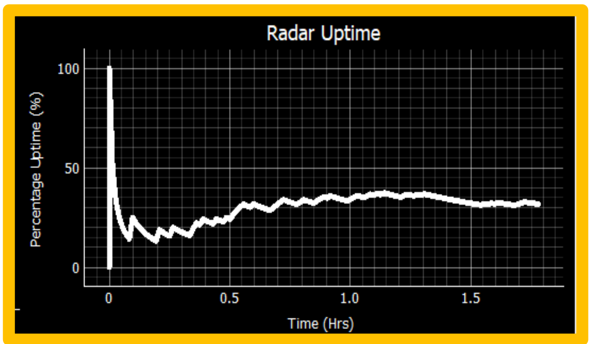
Simulation – Graph View

Satellite Visibility Plot 07



This plot indicates radar visibility over time. Green segments show periods when the radar has a clear line of sight to the satellite, while red segments indicate periods with no visibility.

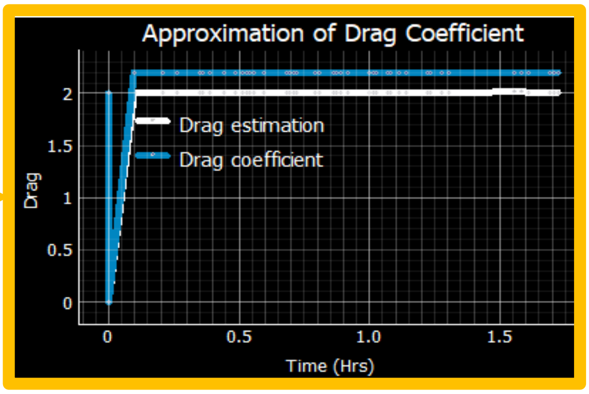
Radar Visibility Ratio Plot 08



This plot shows the percentage of time during which the satellite was visible to the radar. Higher ratios indicate more frequent updates, which typically lead to better filter performance.

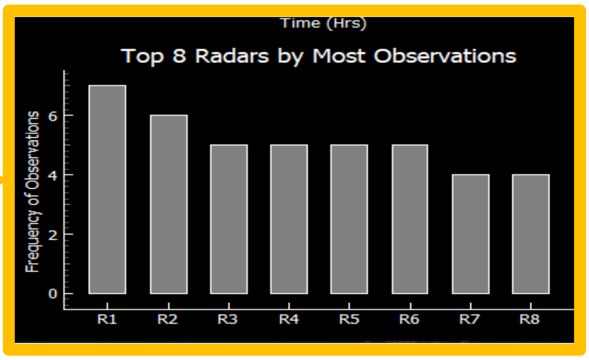
09 Drag Coefficient Estimation

This plot shows the filter's estimate of the drag coefficient over time. The convergence between the two indicates the filter's ability to infer unknown parameters accurately.



10 Radar Observation Count

This plot displays the number of observations made by each radar station. Only the top eight stations with the highest observation counts are shown for clarity.



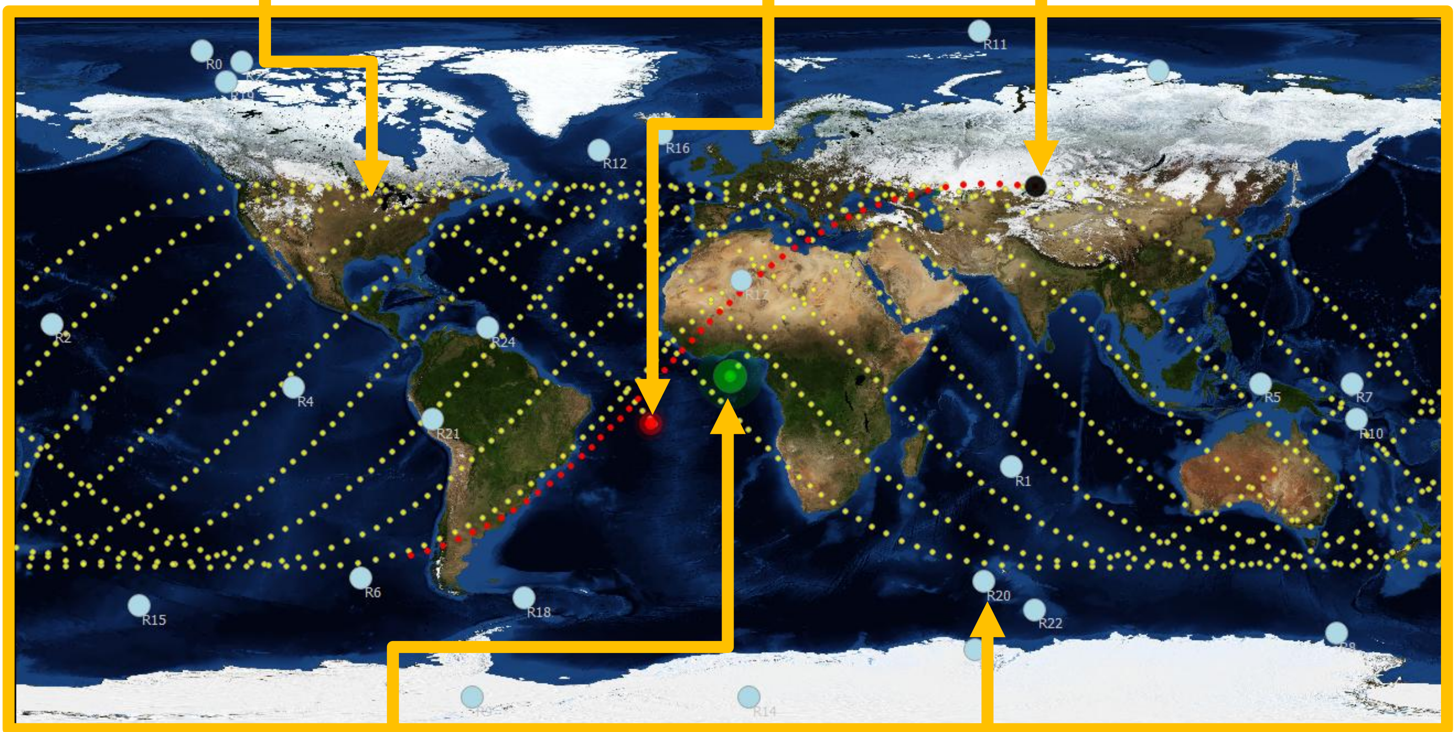
Simulation – Earth View

To switch to the Earth View, click on the “Earth View” button located at the top of the screen. Once selected, the interface will display a global map showing:

01 Yellow dotted lines represent the orbital trajectories.

02 Red dot is the true crash point as per the simulation

03 The satellite is moving along the path in real time



04 The green dot represents the UKF-predicted landing point, which updates automatically as the satellite descends below 140 km altitude

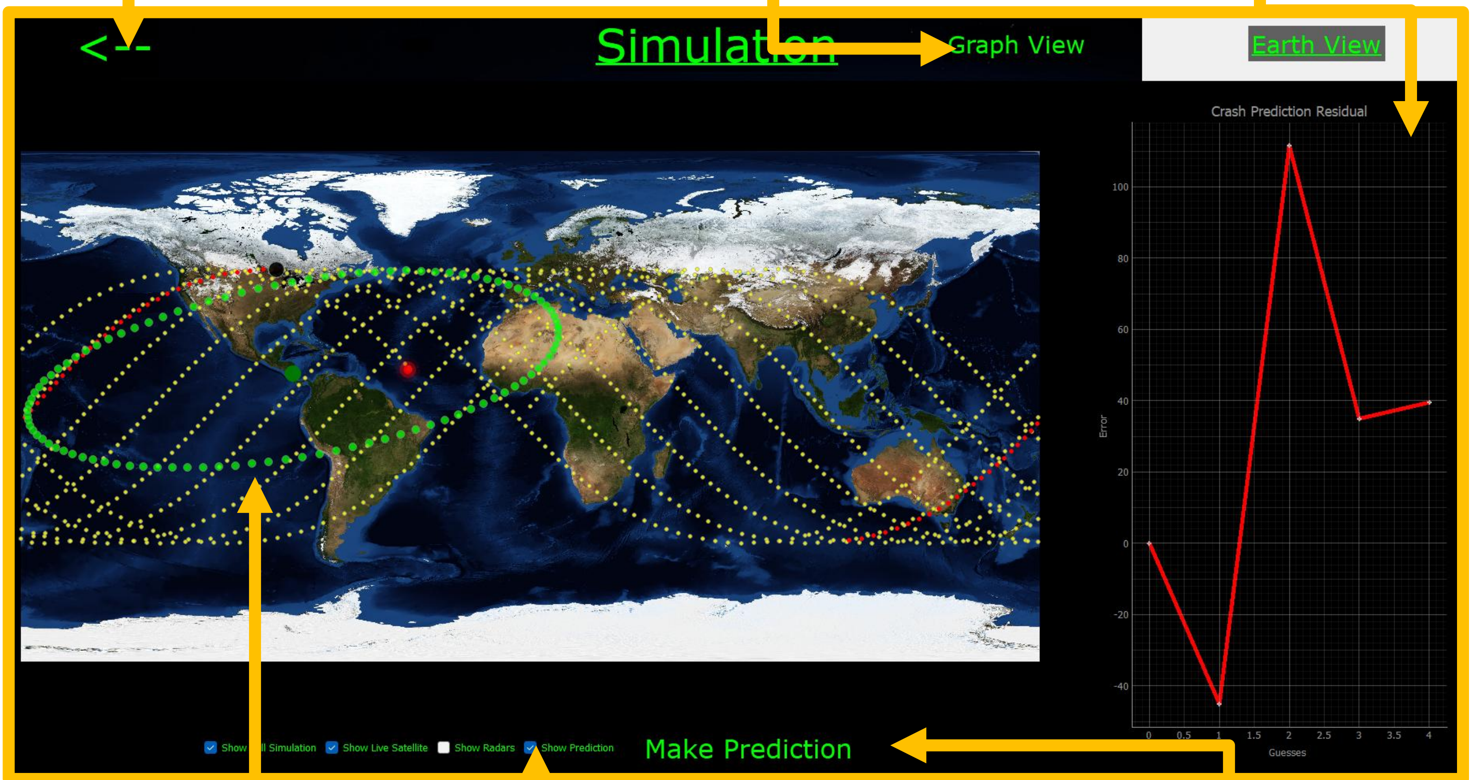
05 Radar sites are shown in light blue in the map. They are hidden by default but can be displayed by ticking the checkbox below.

Simulation – Earth View

06 If you wish to stop the simulation at any point, click the back arrow here.

07 To return to the Graph View, click the button here.

08 This plot will show you the crash prediction residual each time you press “Make Prediction”



09 Once prediction starts, uncertainty region appears around the green dot and gradually shrinks as the estimate improves.

10 You can also toggle the visibility of elements using the checkboxes here.

11 Use the “Make Prediction” button to trigger a landing site prediction during the simulation. It becomes available once the satellite approaches re-entry altitude.

03

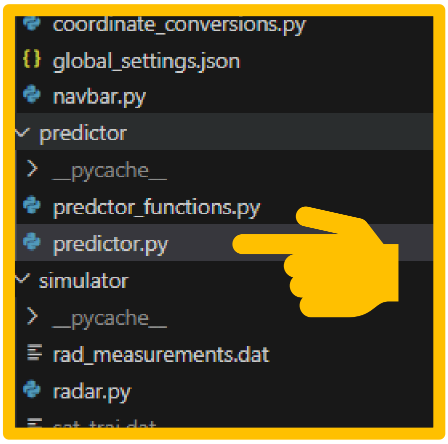
User Configurable Parameters

User Configurable Parameters

In addition to setting the initial position and velocity of the satellite, and selecting the number of radar stations through the user interface, you can further customize the simulation by modifying certain parameters directly in the code.

01 Editing Internal Parameters in Predictor.py

Return to the main project directory, then open the file predictor.py.



1. Adjusting the landing prediction trigger: Navigate through the code until you locate the altitude threshold condition. By default, the value is set to 140 km, which determines when the filter begins estimating the landing position. You can modify this value to start predictions at a different altitude.
2. Modifying process noise configuration: In the same file, you will find the section where the process noise matrix **Q** is defined. Increasing the process noise leads to greater uncertainty in predictions, which may result in wider covariance bounds and more frequent filter corrections.
3. Changing the initial drag coefficient estimate: You can edit the initial value assigned to the drag coefficient Cd . This serves as the UKF's starting guess, and it will be updated during the simulation as new data becomes available.

```
pred_alt = 140e3
```

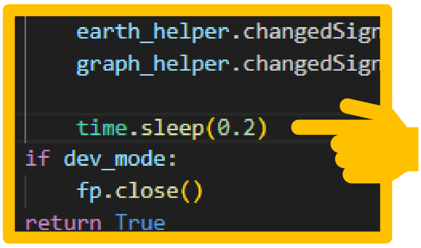
```
### uncertainty in the process model
self.ukf.Q = ukf_Q_7dim(dim=7, dt=dt, var_=0.001, Cd_var=1e-6)
```

```
def __init__(self, grapher, earth, state0, dt=50.0, Cd=2.0):
```

User Configurable Parameters

02 Adjusting Radar Reading Frequency in Radar.py

To modify how frequently the radar provides new measurements, navigate to the radar module inside the project directory.



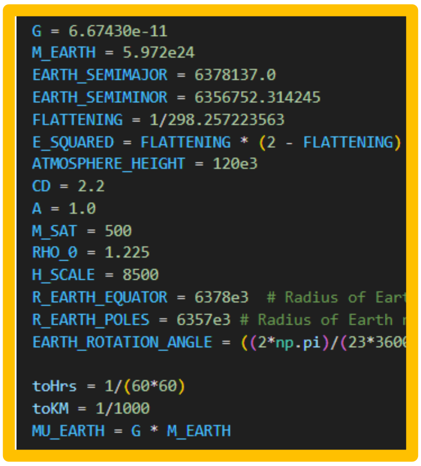
Within the code, locate the line that includes time.sleep(...). This line defines the delay (in seconds) between consecutive radar readings. For example, setting time.sleep(5) means the system will wait 5 seconds before generating the next radar update.

Note: This delay controls the real-world waiting time—not the internal simulation time. Increasing it makes the radar updates appear slower during runtime.

Important: Do not set the value to zero. Doing so will cause the system to overload and crash, as it will continuously request updates without any delay.

03 Adjusting Physical Constants in Constants_trajectory.py

To explore different simulation scenarios, you can navigate to the file constants_trajectory.py, which contains key physical parameters used in the orbital dynamics model. In this file, you may for example modify the mass of the satellite, the cross-sectional area, and the atmospheric model parameters. These changes affect how the simulation behaves and allow you to study different conditions.



Summary

In this tutorial, you learned how to:

- ✓ Set up and run the satellite re-entry simulation
- ✓ Configure initial conditions and radar settings
- ✓ Interpret live visualizations in Graph View and Earth View
- ✓ Modify key parameters in the code for custom experiments

You are now ready to explore further scenarios and test how different conditions affect the filter's performance.

==

Credits and Contact

This project was developed as part of the ES98B Group Project at the University of Warwick.

For questions, feedback, or contributions, please contact the development team.

Team SNOE