ES98B Group Project - Documentation

Group SNOE

May 15, 2025

1 Overview

A satellite is orbiting earth with a decaying orbit. Around its perigee, its orbital motion has started to become affected by atmospheric drag. The satellite is no longer controllable. It is observed by a number of ground-based radar stations, which can get periodic updates on its position and velocity. Aerodynamic drag will cause the satellite to slow down until it impacts the ground. For obvious reasons it would be interesting to predict where this will happen. This software provides a user interface to simulate the re-entry of a satellite using radar measurements and Kalman filtering. The system can be launched using the main script `master.py` after setting up the environment (see tutorial file).

2 Installation Guide

To install the latest version of the software, you will need `git`. If you do not have `git` installed on your machine, please refer to the official installation guide: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Alternatively, you can manually download the repository as a ZIP file from GitHub by clicking on the green `Code` button and selecting `Download ZIP`. Once unzipped, follow the same steps for setting up the Python environment.

Cloning the Repository

First, open your terminal, create a blank folder, and navigate into it:

```
cd your-folder
git clone https://github.com/jrobo-gith/De-Orbiting-Satellite-GroupProj.git
```

Then, navigate into the cloned repository:

```
cd De-Orbiting-Satellite-GroupProj
git pull origin main
```

Creating a Python Environment

In the same folder, create a virtual Python environment:

```
python -m venv venv
```

Activate the environment:

On Windows:

```
.\venv\scripts\activate.bat
```

On macOS:

```
source venv/bin/activate
```

Running the Software

Once the environment is activated, run the following command:

```
python master.py
```

3 User-Facing Documentation

When you run the software:

User Inputs:

- Initial satellite position and velocity
- Number of radar stations (e.g., 50, 100, 500)
- Placement of radars: equatorial or non-equatorial

Usage Notes:

- Landing prediction is only enabled after the satellite drops below 140 km altitude.
- Increasing the number of radars improves accuracy but may increase runtime.
- Radar sites are hidden by default and can be enabled via the checkbox in the Earth View.

2. Developer-Facing Documentation

This project consists of several core components:

3.1 Main Files:

- `predictor.py` – Contains the UKF implementation and all update logic.
- `simulator.py` – Defines the ODE model and solves the trajectory.
- `radar.py` – Simulates radar visibility and measurement noise.
- `visualiser.py` – GUI built with PyQt for interactive simulation control.

3.2 Functions

Each Python module includes inline documentation using docstrings. These describe the purpose, inputs, and outputs of key functions and classes. Developers are encouraged to review the API document to review the functions.

3.3 Performance Tuning

If the user wants to tune some aspects of the model, they could refer to the tutorial for a detailed review of the tunable parameters, but a short list will be provided below:

1. The constants in partials/constants.py containing all the constants used in the model such as satellite mass, cross-sectional area, drag coefficient etc.
2. The rate at which the radars provide observations to the predictor. In radar.py, navigate to the give-radar-observations function, and increase/decrease the time.sleep(...). NOTE, do not remove it entirely as this will cause the loop to give many observations 'at once' which will break the software.
3. Number of samples the predictor takes in predictor.py when we propagate the samples for landing.

3.4 Known Limitations:

The limitations of this code are tied to two aspects - the radar layout, and the computational efficiency of the predictor.

Radar Limitations
Due to low earth orbit being so fast and so low, the radars do not have a large enough field-of-view to make observations of the satellite more than once. This means that for equatorial orbits, the number of radars needs to be around 50-150 for enough coverage such that the code is initialised properly and begins running. For non-equatorial orbits, the number of radars must be 150-500.

Predictor Limitations
As the satellite's trajectory descents below 140km, the predictor begins sampling the posterior and propagates those samples to generate a mean and covariance for the landing site. 20 samples are being generated to predict the landing site which slows down our model considerably. If the user takes more samples to increase the accuracy of the landing site prediction, the performance would decrease to a potentially break the model.

4 Developer Updates

In the future, the user may want to update the following:

- To add a new filter method, extend `predictor.py` with a new class and update logic.
- To change physical constants, modify `constants_trajectory.py`.
- All key parameters are initialised in `master.py`.