Visualiser - De-Orbiting Satellite Predictor

Jack Roberts

April 29, 2025

1 Data visualisation

Data visualisation is the art of representing data such that it maximises the quality to quantity ratio. In other words, display the smallest quantity of data that explains as close to 100 percent of our results as possible. The project proposal, handed in during term 1 outlined the visualiser as a visual representation of the predictor (Extended Kalman Filter) and will display temporal as well as spatial data regarding the trajectory of the satellite. There are many data visualisation tools to accomplish this, such as Unity (C#), matplotlib (Python), pyQtGraph (Python) or Tableau.

2 Methods

2.1 A change in software

Unity and C# was the software planned for use for the visualiser. It had a built in Graphical User Interface (GUI) and was able to convert large amounts of data into highly customisable graphs. Upon reflection, and some guidance from the academics, it became apparent that using as many as three languages to complete this project was too much. Moreover, the transfer of data was proving to be difficult as running a python venv inside of C# and converting its output to C# arrays was a significant bottleneck. Instead a Pythonic approach became the apparent option.
The visualiser now runs inside python, running the *pyQtGraph* module. This module builds on its parent module *pyQt* for creating python GUIs by adding graphical support such as plotting, adding legends, multiple subplots and dynamic plotting, where the figure could update as it receives updates from the predictor in real time. This was an excellent choice as this module allowed the project to be kept in python, a language everybody in the group understood, without sacrificing dynamic updates from our predictor in real time which is what was needed.

2.2 Installation instructions

2.3 Classes

There are two classes, a **MainWindow** class and a **Plot** class. **MainWindow** runs the GUI, keeping track of plots. It instantiates the GUI by setting a graphical layout unit and window size, it instantiates the subplots chosen prior to its instantiation using the **Plot** class. The **Plot** class is responsible for each subplot. It takes initial data matrices

(numpy arrays) x and y to begin the plot. The dimensionality of x and y should be $M \times N$ where M is the number of lines and N is the number of observations.
Plot takes as inputs:

- Its allocated subplot,
- Initial x,
- Initial y,
- Args ,

Where *Args* is a dictionary of local information that specific plot needs to know, such as:

- Title,
- Label X title,
- Label Y title,
- Legend line names,
- Legend (Bool),
- Grid (Bool),
- Line details (line width, colour etc),
- Symbols (eg. each data point represented as a '+'),
- Label style,
- Title style

Arg profiles are loaded when a **MainWindow** class is instantiated. Each plot has its respective *profile* encased in a JSON file, and is used for styling the plot.

Within the **Plot** class, there is an *update* function. This allows for appending new data given by the predictor to the existing array and plots it. It removes the oldest addition to the array and appends the new value.

3 Plots

There will be IDK plots, each displaying vital information representing the current state of both the satellite and the Kalman Filter's prediction of its predicted landing site.

4 Results