Ehtesamul Azim (UCFID: 5460629)
Wally Proenza (UCFID:5456554)

Nonlinear Systems
Five UAV Cluster SMC Project
Professor Enyioha
12/08/2023

Five UAV Cluster Sliding Mode Controller (SMC) Final Project

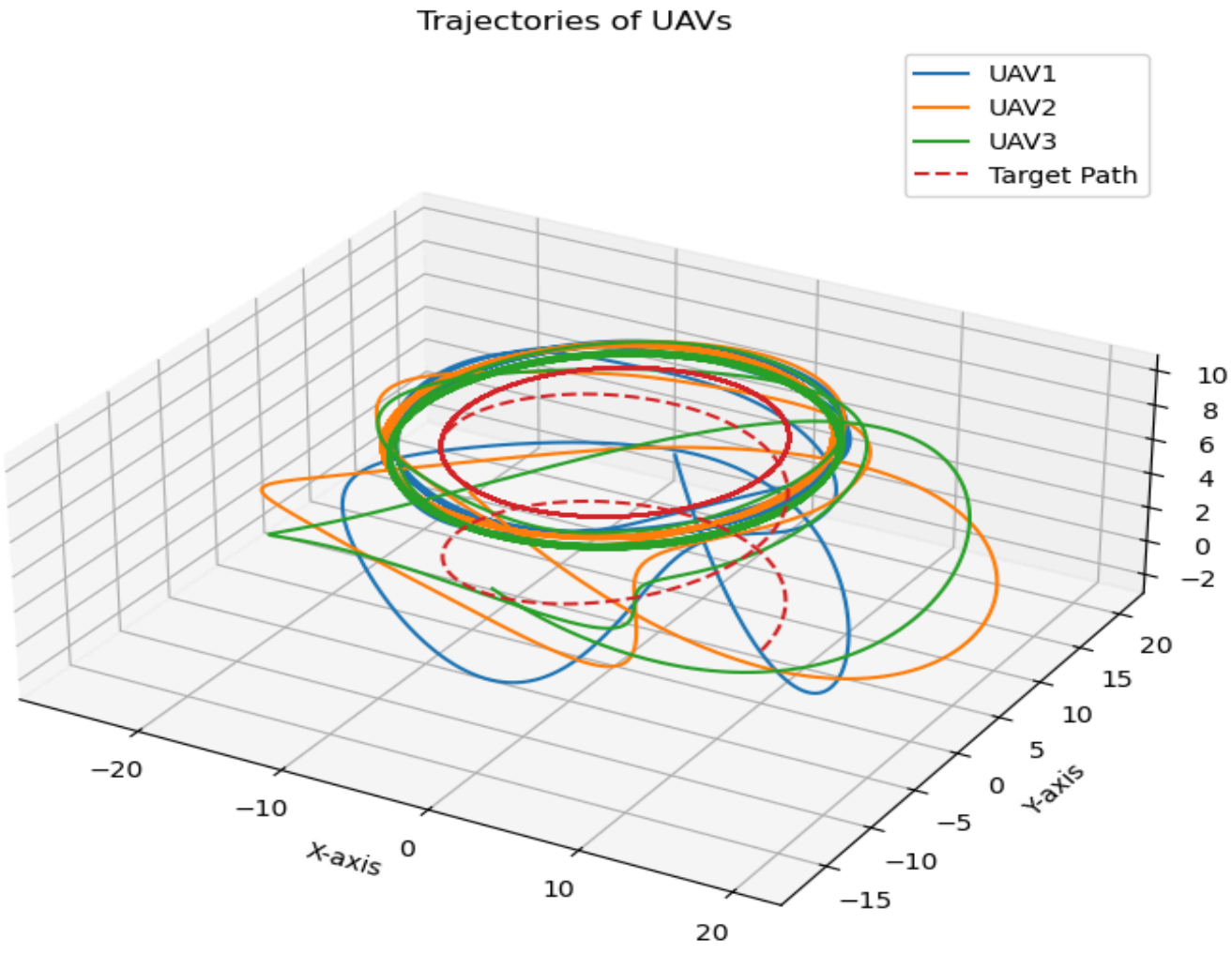
Preface:

Two different implementations of the five UAV SMC were explored: One in MATLAB Simulink (5 UAV implementation) and one in Python (3 UAV implementation). Both implementations could use further refinement but function as a suitable starting point for future independent study on the subject outside of a classroom environment. The Simulink implementation and Python implementation will be briefly discussed within these pages.

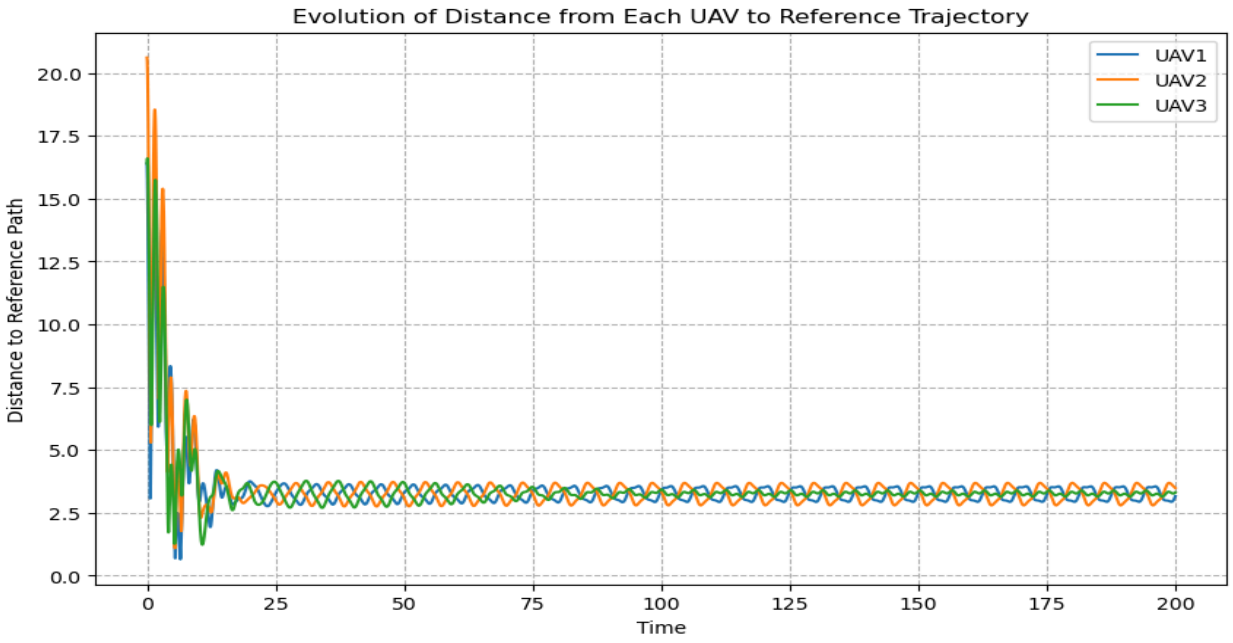
For better view of the code and the simulink stuff (open UAV_SMC.zip for Simulink Files), please check our github repo (check the main_code.ipynb for this project's code and if you wanna see our effort to learn to code sliding mode control please check the other files :'))

Using Python:

I will be including the results for 3 UAVs here but because of time constraints we were not unfortunately able to finish the python implementation for 5 UAVs. Also I am not sure how to get the 3d view like the one in the paper, and weirdly all the results in our code are basically flipped off the ones you see in the paper. Not sure why. I could have manually changed it but that would be cheating. **The following figure shows the trajectory of the 3 UAVs with no perturbation and no gps error.**



As you can see, in the beginning, as the UAVs are pretty far from the trajectory center (0,0,0) they take some time to figure out their destination **but once they get closer to the target, they converge pretty well and their stability is shown in the following figure:**



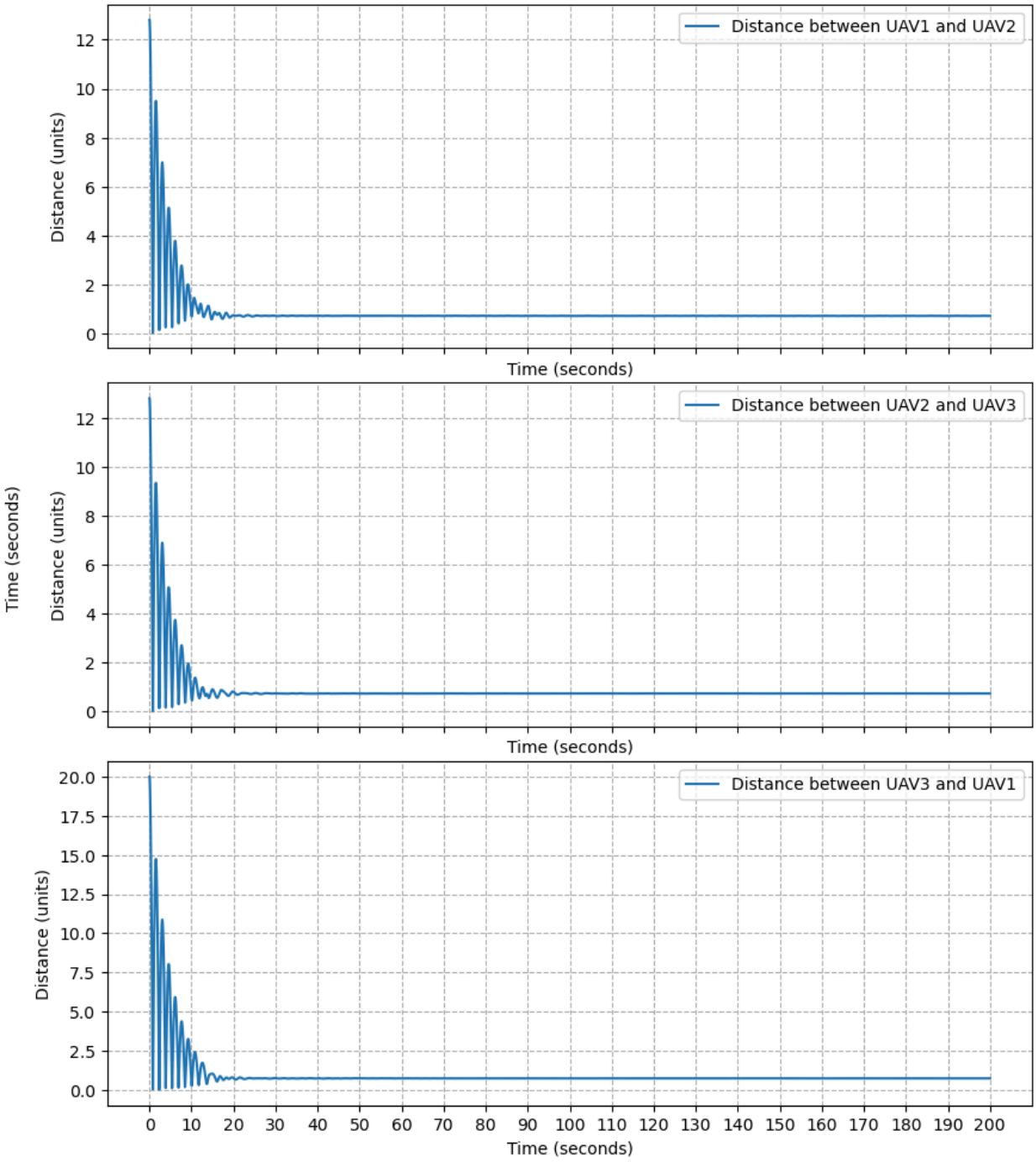
The figure in the next page shows the effectiveness of the repulsion force between the UAVs.

Thing I could not finish: Checking performance after adding perturbation, I added perturbation in my function as follow:

```
# Perturbation forces
    fd_x1 = 10 + 10 * np.sin(0.2 * np.pi * (t - 80)) * (t >= 80)
    fd_y1 = 5 + 5 * np.sin(0.2 * np.pi * (t - 140)) * (t >= 140)

    fd_x2 = 10 + 10 * np.sin(0.2 * np.pi * (t - 80)) * (t >= 80)
    fd_y2 = 5 + 5 * np.sin(0.2 * np.pi * (t - 140)) * (t >= 140)

    fd_x3 = 10 + 10 * np.sin(0.2 * np.pi * (t - 80)) * (t >= 80)
    fd_y3 = 5 + 5 * np.sin(0.2 * np.pi * (t - 140)) * (t >= 140)

    fd_z1, fd_z2, fd_z3 = 0, 0, 0
```

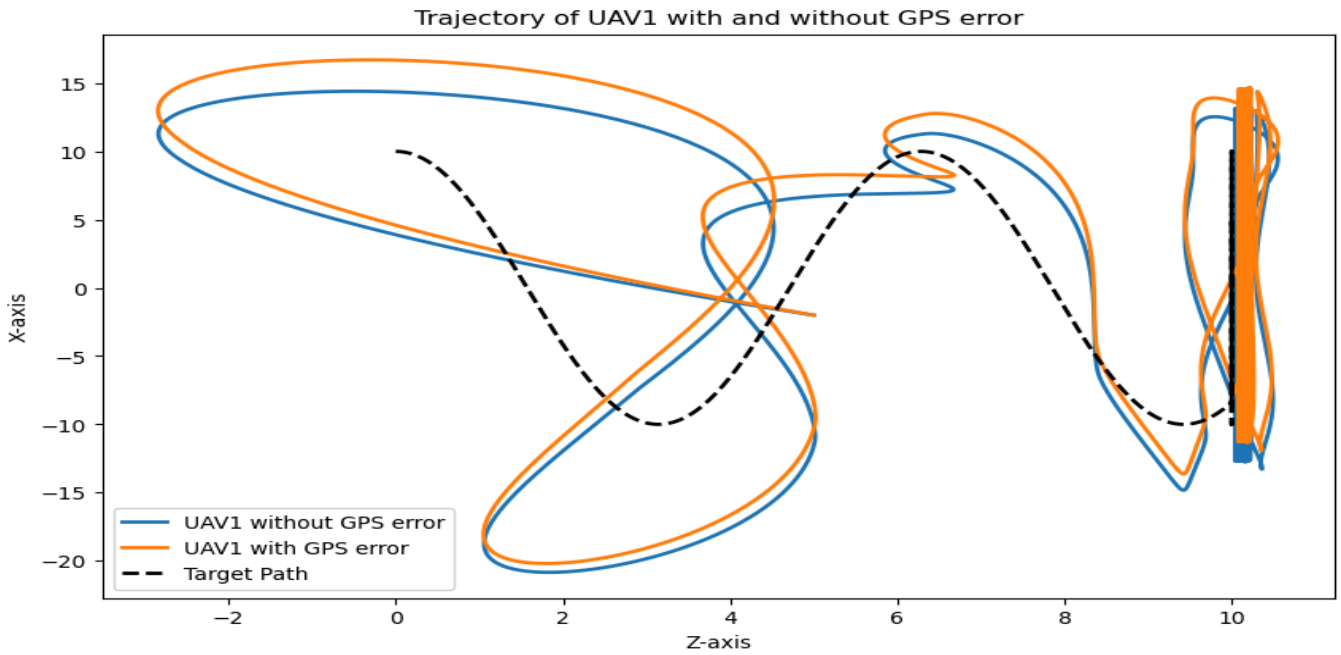
But due to time constraints again, I could not debug the performance glitch I was getting.

Evolution of Distance Between Each Pair of Robots

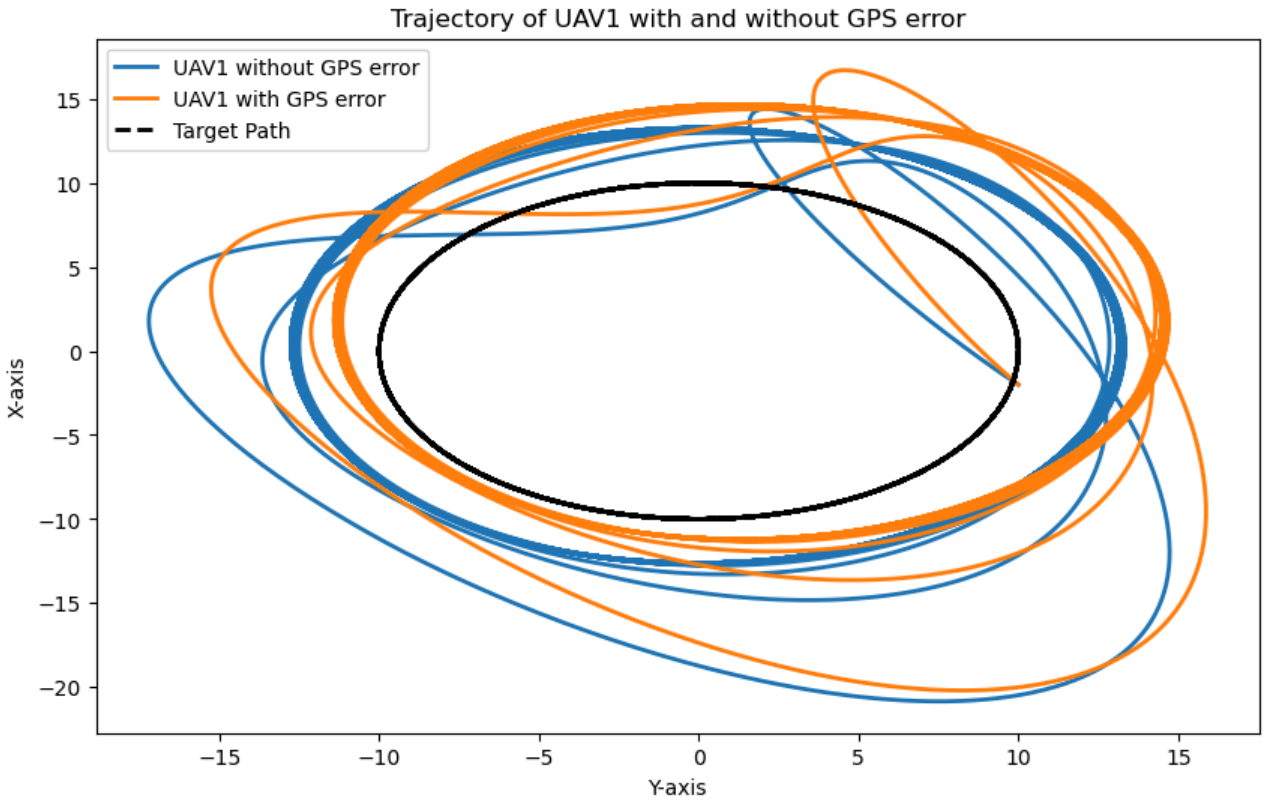


Adding GPS Error:

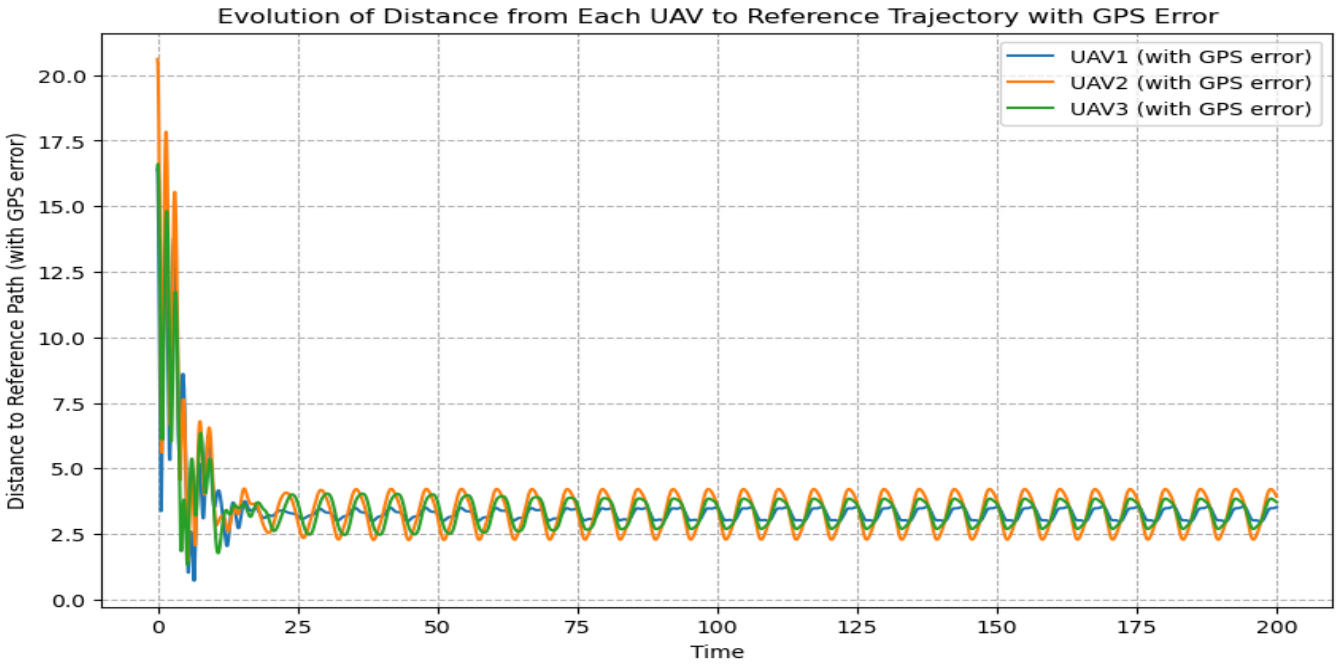
Our implementation works fine even after adding GPS error (+-3m of error in reading) as shown in the following figure:



This shows that as time goes by, UAV1 catches up with the target path even after having errors in input reading/gps error. Similar thing can be seen if you try for Y and X axis:



And all 3 UAVs converge pretty well even after having GPS error as shown in the following figure:



Things I would have liked to figure out:

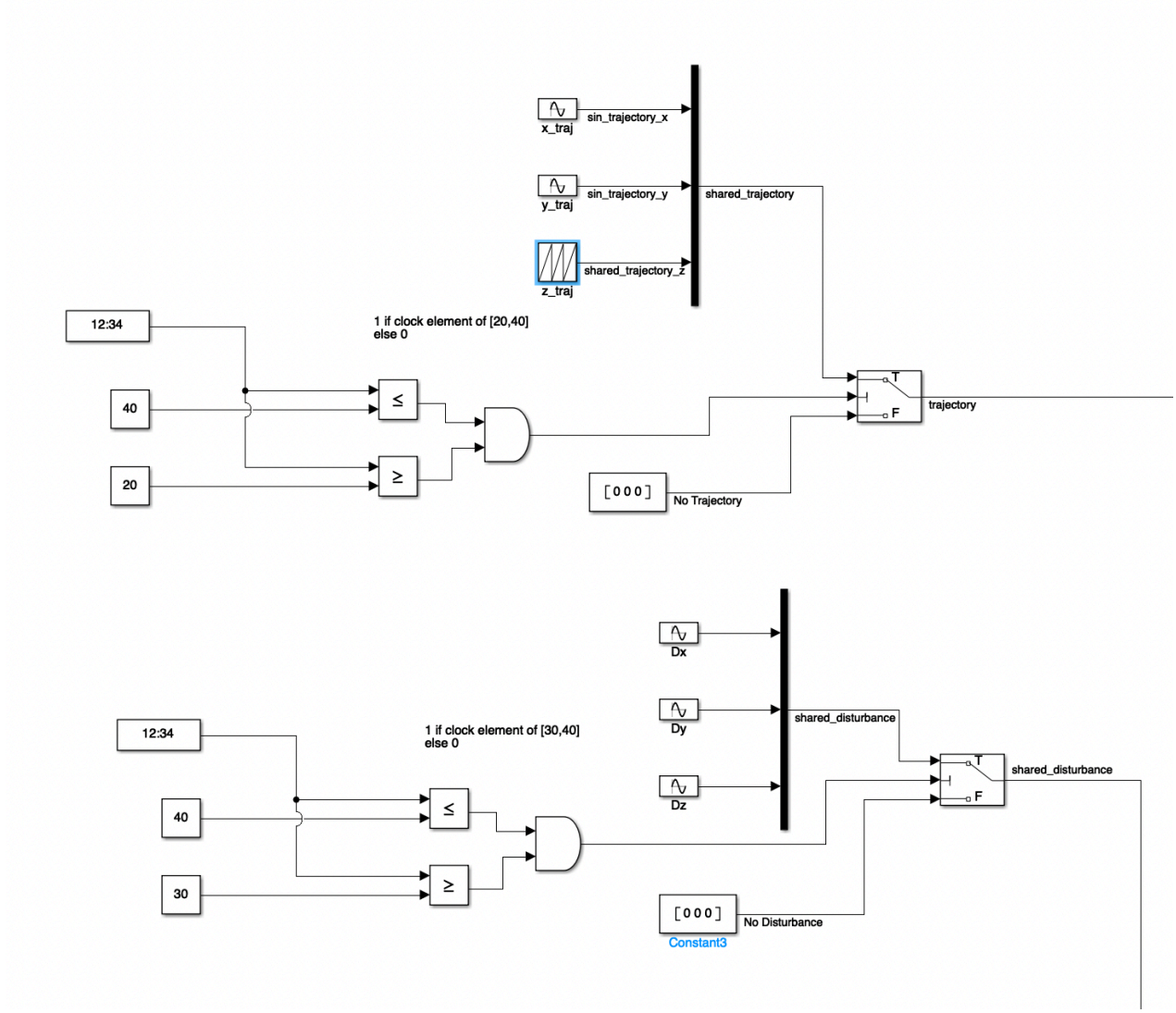
1. ***Why was the perturbation not affecting the performance of the UAVs***
2. ***Not bothered so much about the 5 UAV implementation because it's a straight up extension but unfortunately we were in a rush.***

Simulink Model:

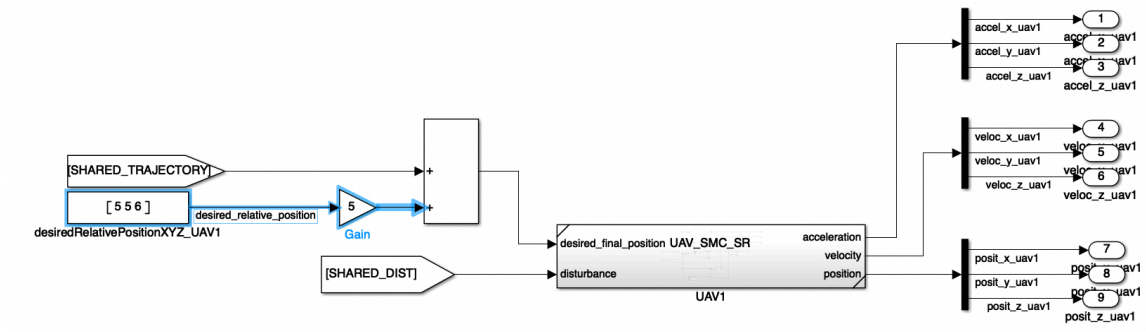
To use the Simulink Model simply open MATLAB 2021b, open the UAV_SMC.prj, wait for everything to load, and Run the UAV_CLUSTER_MAIN.slx

Below is the file dependency hierarchy:

The Simulink Model takes in a desired trajectory and a disturbance that represents a noise. Due to time constraints, the UAVs are located within a sphere from the start of the simulation and move in sync with each other throughout the simulation. For simplicity the same sinusoidal noise was applied to all x, y, and z axis of the UAVs. For the purposes of this experiment the input trajectory, from t = 20 to t = 40 in the simulation, were sinusoidal for the x and y axis, and strictly increasing for the z axis. However, the input trajectory is reduced to 0 before t = 20 and after t = 40. Meaning that while t is less than 20 or t is greater than 40, the UAVs will travel back to the originally desired locations. This is discussed further in the next paragraph.



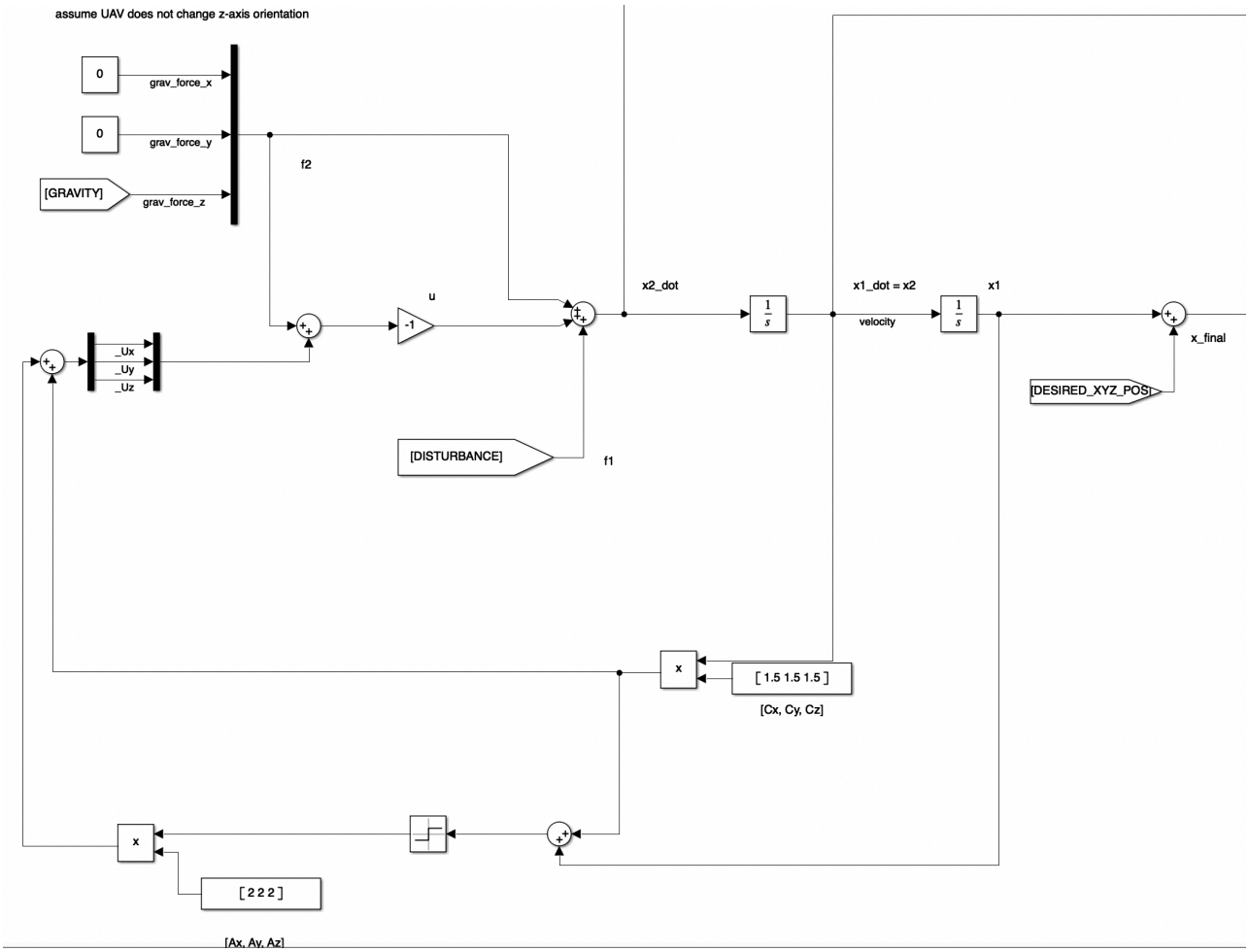
The UAVs have the same initial conditions, except for the fact that they are spaced away from each other within a sphere. The UAVs will converge to their preset location which at this time is not a user input but instead set within the second layer of the model. A gain is used to increase or decrease the spacing between the UAVs, this gain is not a user input but it is located within the second layer of the model and can be altered there:



The image above shows that the original desired position of UAV1 is equal to [5 5 6] multiplied by a gain of 5 to achieve [25, 25, 30]

Similarly:
The original desired location for UAV1 is: X = 25, Y = 25, Z = 30
The original desired location for UAV2 is: X = 30, Y = 25, Z = 25
The original desired location for UAV3 is: X = 25, Y = 30, Z = 25
The original desired location for UAV4 is: X = 20, Y = 25, Z = 25
The original desired location for UAV5 is: X = 25, Y = 20, Z = 25

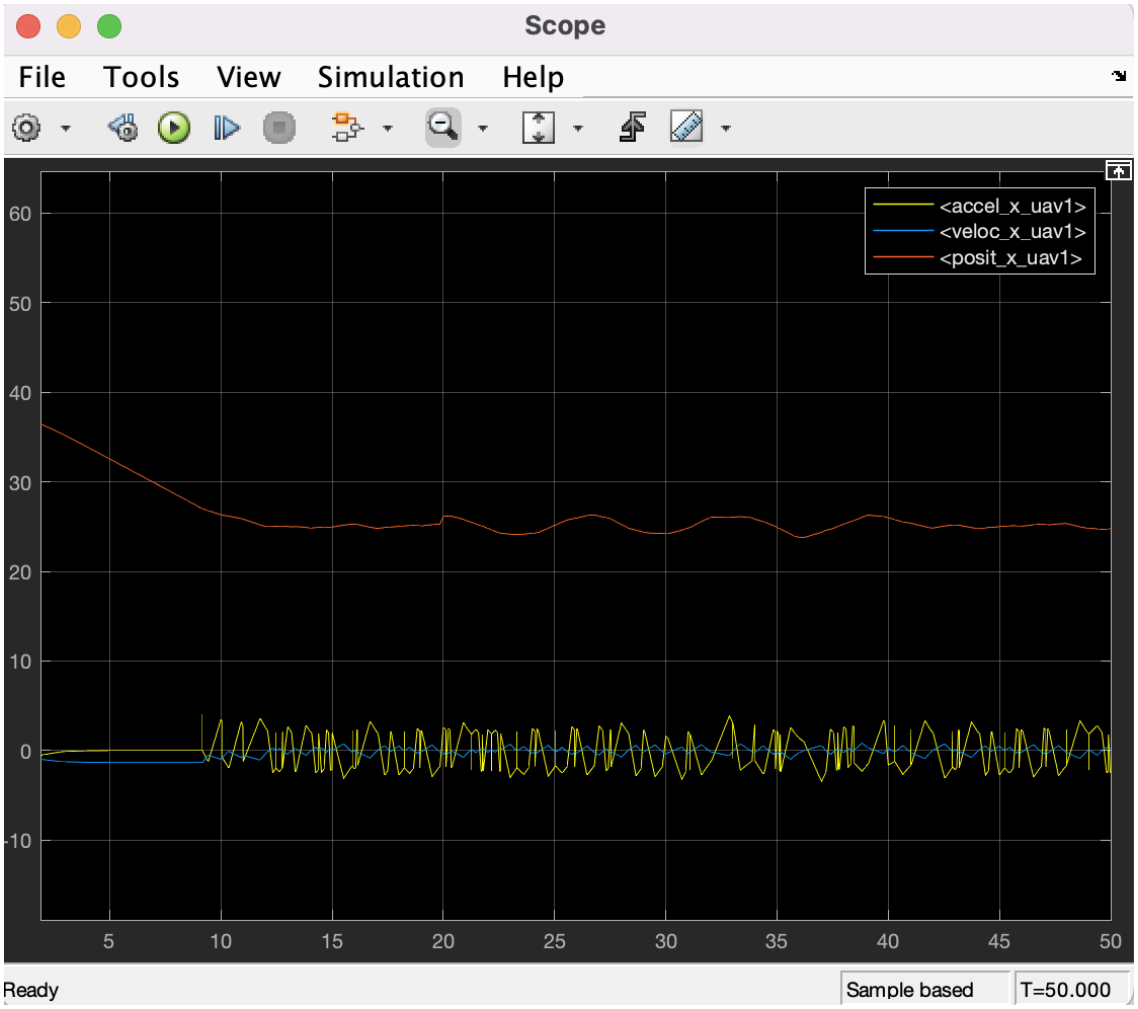
The system equations are modeled by the sliding mode controller shown below:



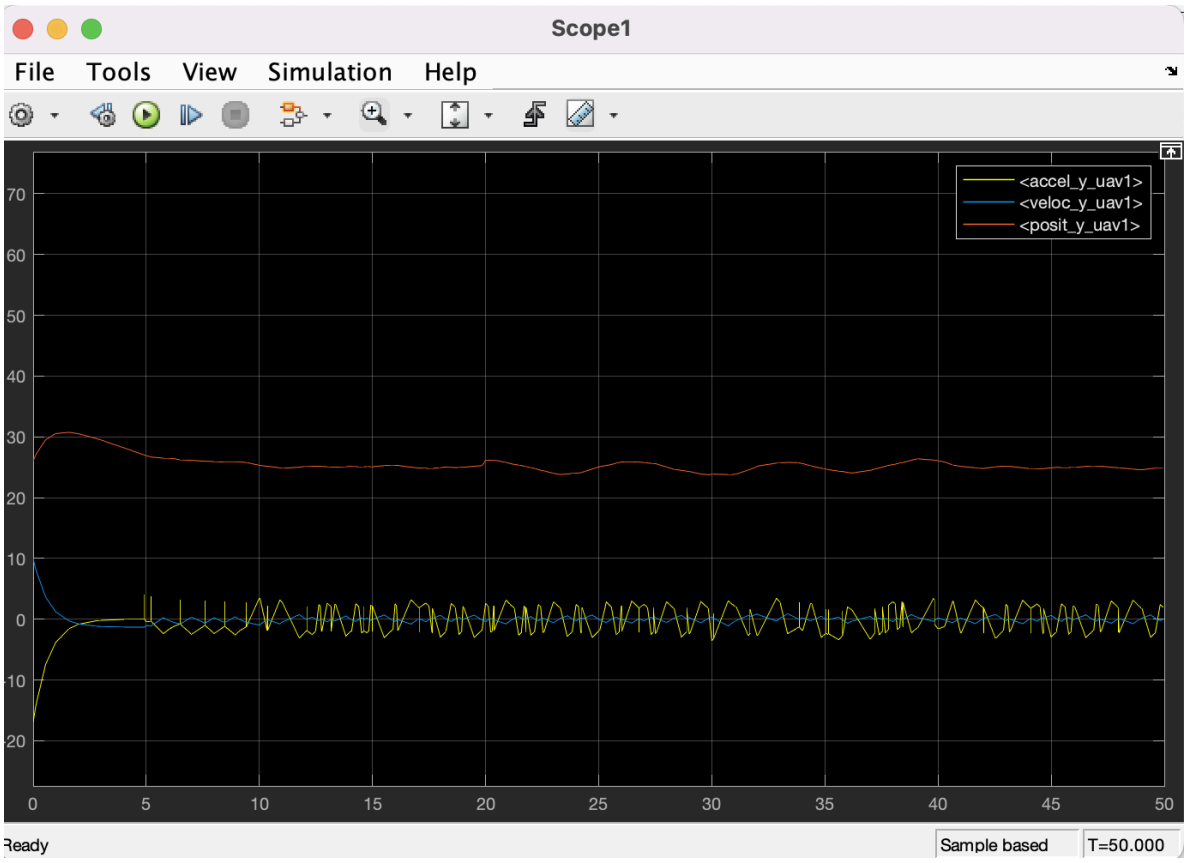
x_final = x1 + (desired xyz position)
desired xyz position = (top level user input trajectory) + (originally desired UAV location)
x1_dot = x2
x2_dot = u + f
f = f1 + f2
f1 = top level user input disturbance
f2 = the gravitational pull on the UAV x,y,z axis. For this model the UAV is assumed to never change it's z-axis orientation. As such only the z-axis experiences the -9.8 gravitational force f2.
u = - (c * x2 + a * signum(x1 + c * x2) + f2)
a = [2, 2, 2] reaching time parameter constant
c = [1.5, 1.5, 1.5] manipulates the asymptotic convergence

Note: All variables described above are 3-vectors representing the x, y, and z axis of the respective UAV.

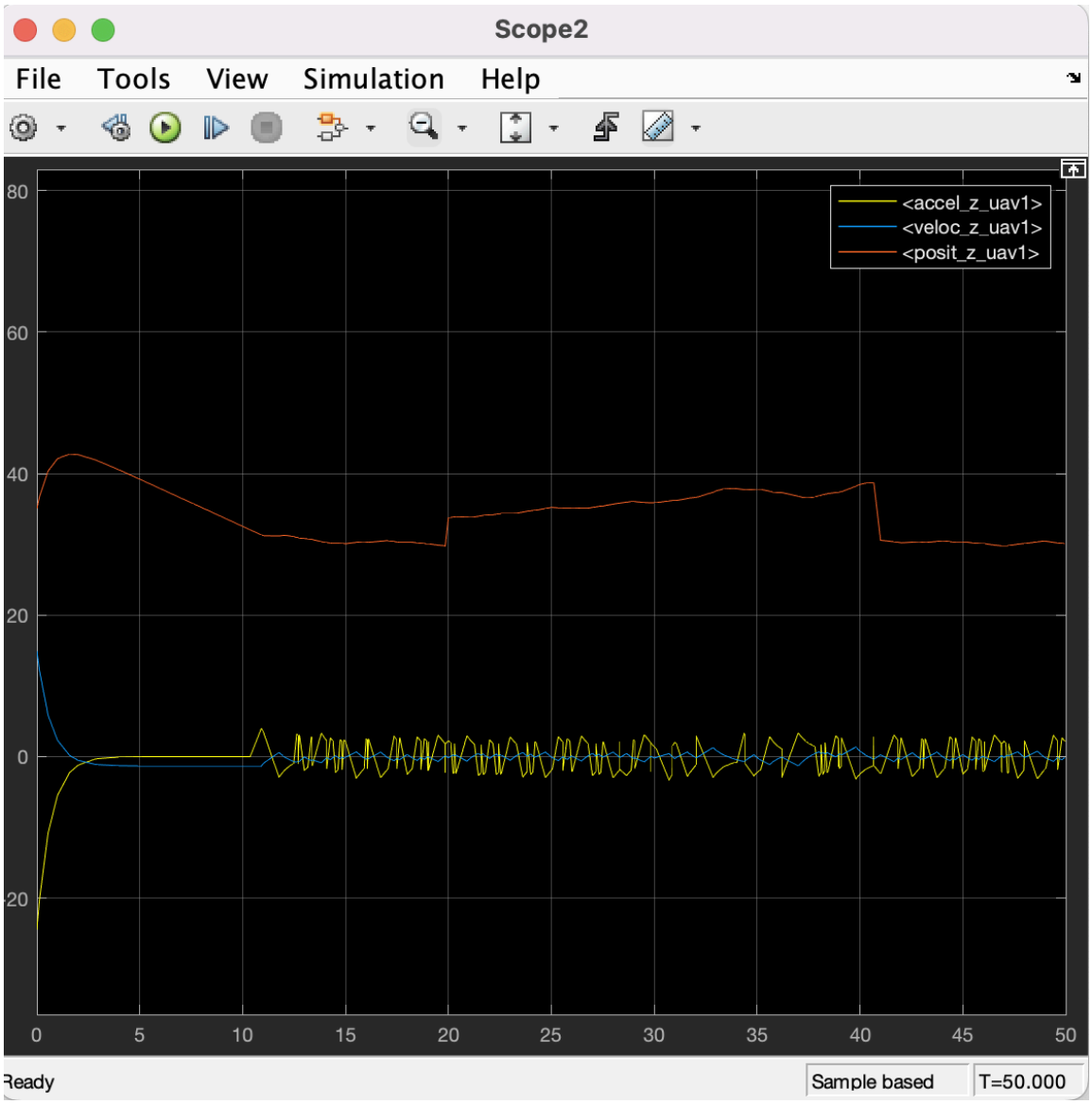
UAV1 X-Axis Graph (obtained from top level scope):



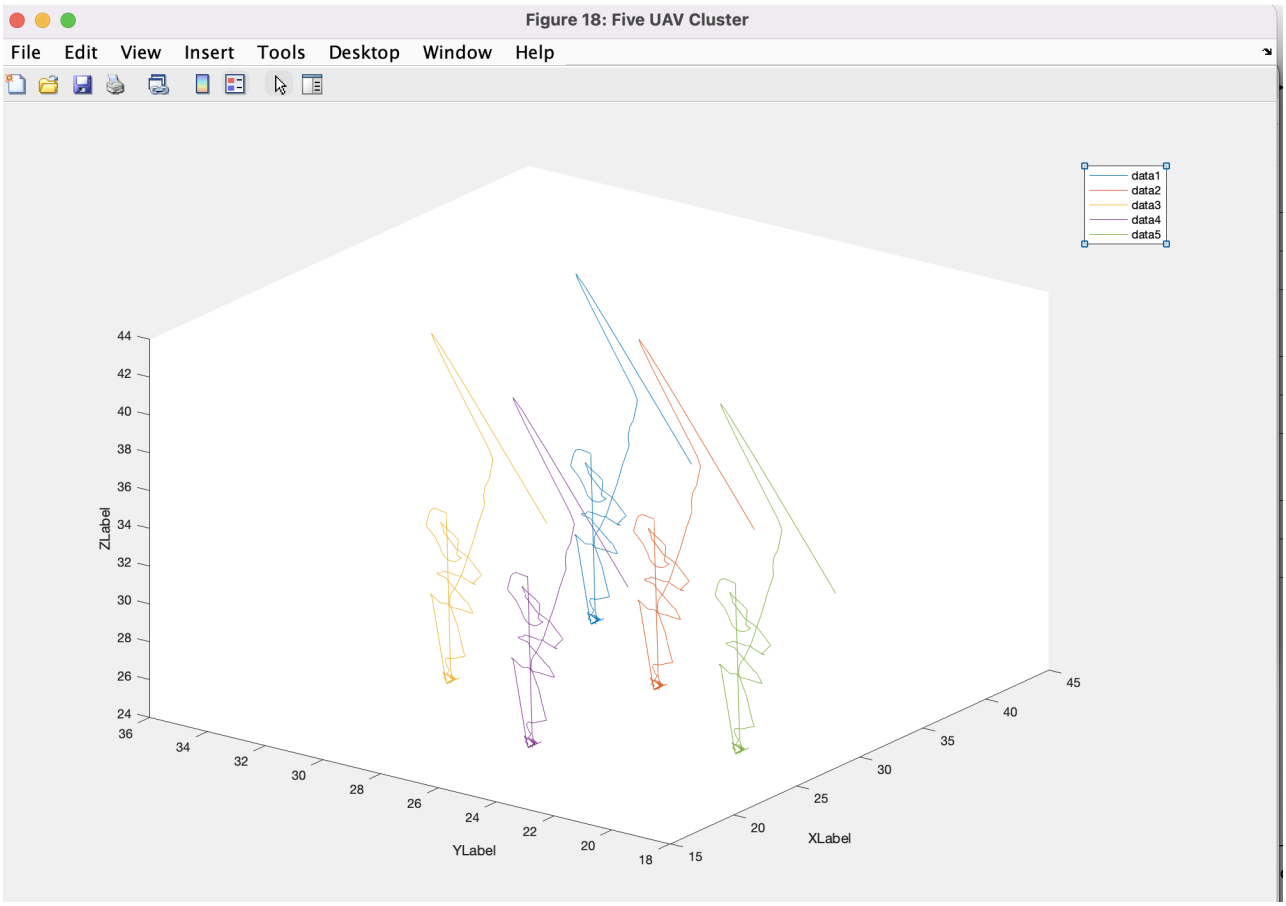
UAV1 Y-Axis Graph (obtained from top level scope):



UAV1 Z-Axis Graph (obtained from top level scope):



UAV2, UAV3, UAV4, and UAV5 all have similar graphs to that of UAV1. The XYZ trajectory of all UAVs is displayed below:



Note: data1 = UAV1, data2 = UAV2, data3 = UAV3, data4 = UAV4, data5 = UAV5

From time interval [0, 20) the UAVs have no input trajectory and settle into the original desired locations. From time interval [20, 40] the UAVs receive input trajectory and follow the trajectory. From time interval (40, 50] the UAVs return to the original desired location.

References:

- Nonlinear control for collision-free navigation of UAV fleet
  by Alexander Martinez Alvarez and Carlos Alberto Lozano Espinosa
- Sliding Mode Control Youtube Playlist by @intuitivecontrolsystem4436
  https://www.youtube.com/playlist?list=PLIF9ObQIYUSeCn2i7hzAZRh8RSb2bObhK