Week 3 Report - Dec 5, 2022 ~ Dec 11, 2022

Goals

1. Implement the NPFG logic in PX4 library in the OpenAI Gym WindyWings (Private Repository): https://github.com/Jaeyoung-Lim/windywings-gym.
2. Evaluate how the Multicopter vs FixedWing behaves, when the airspeed reference is set to 0 (stop when on-track with the path setpoint)

WindyWings Repository

Was setting the repository up in the Windows environment, and created PR for building missing dependencies when using Python 3.11. But overall it was a marginal progress, as Windows is just not as good a dev environment as Ubuntu (packages, software installs, user privileges).

To get up to speed with the 'windywings-gym' library, read and got concepts of the Gym environment for the first half of the week: https://www.gymlibrary.dev/

Read and learned about the Pygame module, on which functions are in charge of rendering the screen, as creating a 'Wrapper' for the Fixed-Wing Lateral Acceleration command environment and drawing the desired path over in the wrapper `render` function wasn't working properly.

Figured out that by using the concept of `rendering_mode`, I could get the raw RGB array and overdraw on that and display it to the user in the Wrapper, but thought that would take more time and decided to create a hard copy of the FixedWing Lateral Acceleration environment specifically for NPFG, and edit the function `render` inside directly.

NPFG Implementation

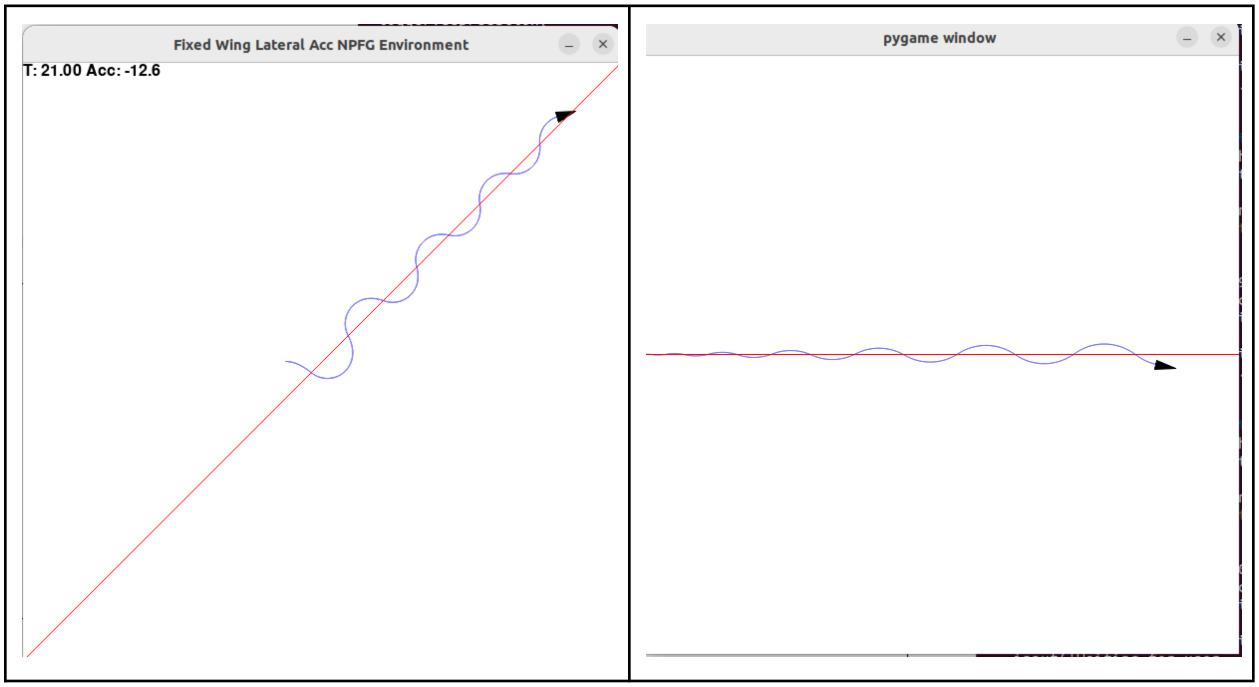
- Implemented the core logic on lateral acceleration based on path following conditions (Path setpoint, Path unit tangent vector, Ground velocity, Position) in `NPFG.navigatePathTangent_nowind` function.
- Didn't implement Airspeed regulation logic yet (longitudinal acceleration)
- PR (won't be merged) for visibility is here.

Notes:

1. **Frame of Reference**: I decided to use the XYZ convention to Right-handed, East-North-Up & Right-Front-Up.
    a. This has complications in NPFG logic (as to where the 'lateral acceleration' points to: Y-axis)
    b. Also, since PX4 uses the Right-handed North-East-Down & Front-Right-Down frame of reference.
    c. The reason was to keep the visual representation of the vehicle in simulation to be consistent with Pygame's XY graph concept
    d. Now that I reflect on it, I could have just used the PX4 convention (which equals to Aerospace literature convention), and just flipped the Pygame display to 'x = y' axis 🤦

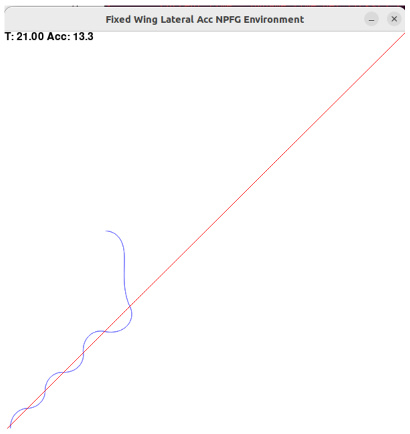
Simulation results

Fixed Wing, Path bearing = PI/4	Fixed Wing, Path bearing = 0



Notes:

1. **Vehicle doesn’t follow the path correctly when it’s not on path in the beginning**
    a. Example: When the path is in PI/4 (up-right direction, since X-axis is horizontal and Y-axis is vertical), and vehicle starts in far-left position, the vehicle goes in -¾ PI direction instead. The bearing turning logic is broken.



2. There are lots of oscillations in the path following behavior
    a. Acceleration command seems to be saturating at +/- 13 m/s^2, which is p_gain (=0.8) * airspeed (=15), which means that heading error is more than PI/2 in magnitude. This is odd as it seems that vehicle is always within ~PI/4 heading error range.

Retrospective

- I tried to refine the code extensively to make the code ‘clean’ (architectured), but this prevented me from actually working on NPFG library implementation more. Since Saturday I took on the ‘Make dirty and Break things’ approach, and this seems to work better for now at least! It improved the development speed significantly 👍

Misc

- Set up a new Ubuntu dev setup to avoid problems arising from Windows environment on Wednesday (Dec 7, 2022). Definitely accelerated the development process.