Week 4 Report - Dec 12, 2022 ~ Dec 18, 2022

Note: Weekly meeting was held on Dec 14, 2022 (Wednesday).

Goals

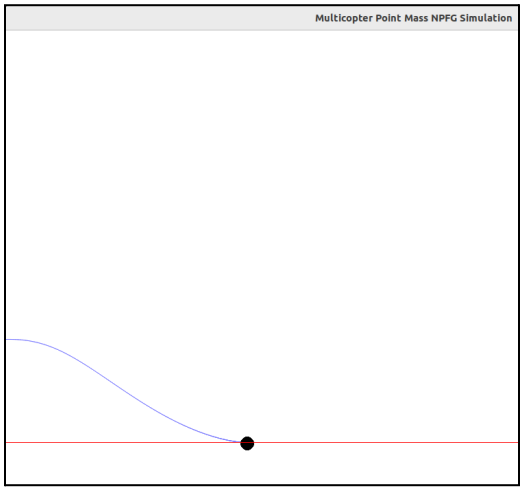
1. Test how having nominal airspeed to 0 and using air velocity reference for guidance works on a multicopter model
2. Apply to Multicopter & Fixed-wing and think about the differences in control mechanism
3. Try out different ramp-in/ramp-out functions to manipulate the air-velocity reference vector, breaking it down to a component parallel/orthogonal to the unit-path-tangent vector

Results

1. Air-velocity reference based path following works for point-mass multicopter model
2. Track keeping feature has limitations on commanding it's maximum 'minimum ground speed' (driving vehicle to path), due to the `bearingFeasibility` function, this logic needs to be revisited.
3. Haven't tried different ramp-in functions yet

Multicopter point-mass model

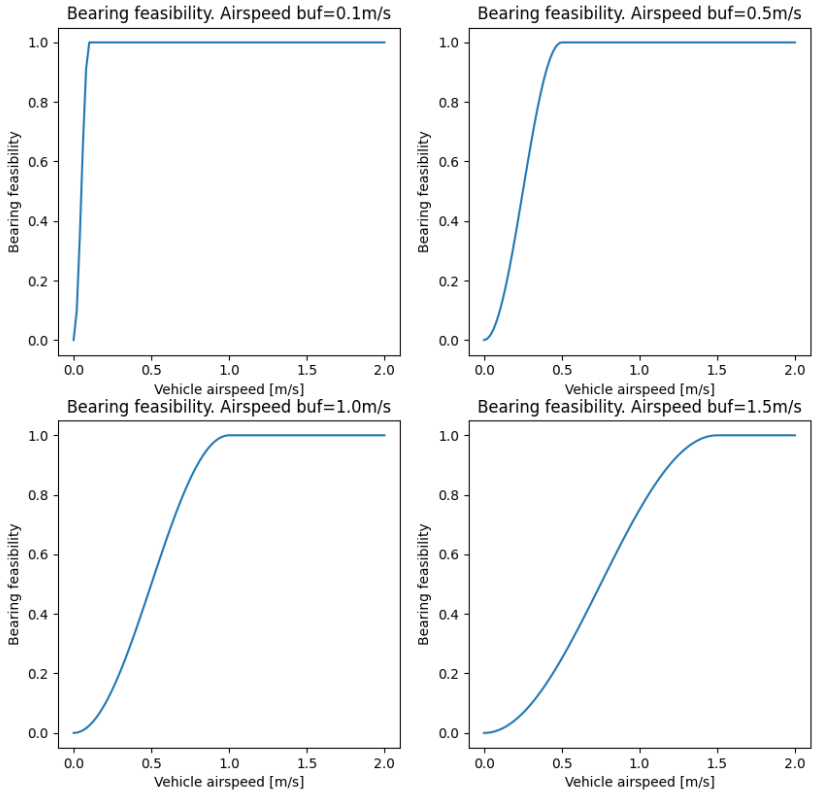
A point-mass environment of a multicopter with a P-controller for velocity has been constructed. It was commanded to follow the air velocity reference vector as an input, which resulted in a curve shaped like this (with user-set minimum ground speed = 5 m/s, track keeping max min ground speed = 5m/s):



Since the nominal airspeed is 0 (on the path, vehicle will come to a stop), the Multicopter comes to halt as it approaches the path.

Findings:

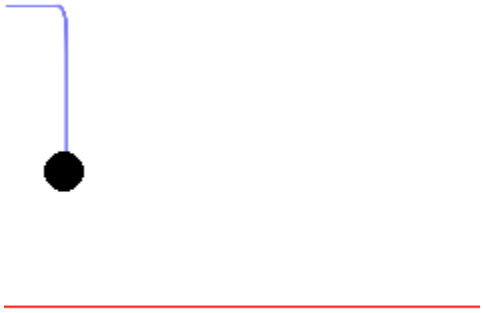
1. **Native 'track keeping' feature only comes into effect when the bearing feasibility is less than 1.0** (not fully feasible), as commented here.
2. **User set 'minimum ground speed' needs to be 0.0 when the track keeping is active**, according to the PX4 Dev Summit Presentation.
    a. But why? Why can't both 'user-set min ground spd' & 'track keeping based min ground spd' come into effect?
3. **Implementing the `bearingFeasibility` function (previously not implemented) properly is necessary for track-keeping feature to work**, due to point 1.
4. When the vehicle's current air velocity is not 0, the bearing feasibility sharply rises to 1.0 as it crosses the 'airspeed buffer' value.



<u>As shown above, the bearing feasibility function relies greatly on the ‘airspeed buffer’ defined in NPFG</u> (Hard-coded as 1.5 m/s, in PX4 library).

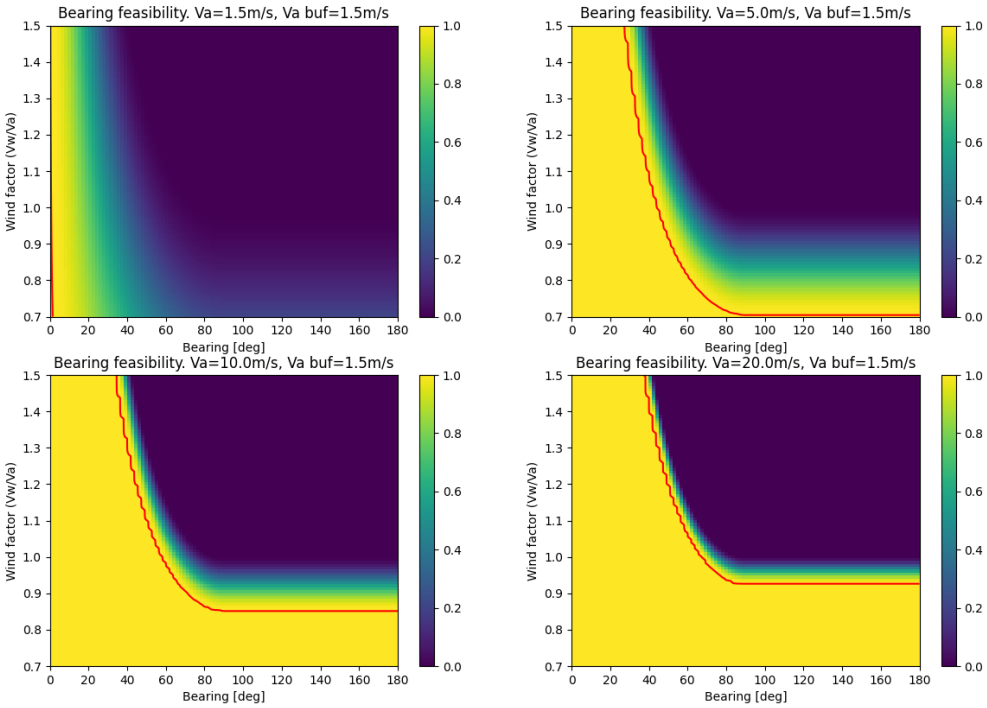
This relates to the point 4 above, where for a multicopter with nominal airspeed set to 0.0, as soon as it picks up minimum ground speed from track-keeping feature, it will reach an equilibrium where the feasibility is somewhere between 0.0 and 1.0, and the resulting minimum ground speed from track-keeping feature sets the stable vehicle airspeed.

In other words, when nominal airspeed is 0.0, track-keeping induced minimum ground speed can never exceed ‘airspeed buffer’ value.



With nominal airspeed = 0.0, user-set minimum ground speed = 0.0, track-keeping maximum minimum-ground speed = 5.0, the <u>vehicle reached steady state with ground speed of 1.15 m/s, with a combined feasibility of 0.75.</u>

Bearing feasibility



‘Bearing feasibility’ is a key function that allows setting feasibility to 0.0 that enables having a valid minimum ground speed set by the track-keeping feature. To test the functionality & get an intuitive feeling of the function’s behavior, I have graphed it using the native NPFG library implemented in the environment.

Retrospective

In general, it seems like it took me a full 4-weeks with trial & error of implementing NPFG to understand it quite well (at least I think so). Small details like bearing Feasibility function & track keeping ground speed manipulation, etc has so much more complexity than I thought the week before, and I am finding myself having the NPFG python version closely assimilating the C++ library version at this point.

However, since I didn’t do much of a quantitative analysis on the path following ability, progress in this needs to be made next week without a doubt.

Air velocity reference control

After the weekly meeting I realized that the final control doesn’t have to be an ‘acceleration’ command (and this was discussed in last meeting on Dec 5, 2022 apparently), but I can directly use the air mass relative velocity setpoint vector for control!

Gym environment

After having a delayed weekly meeting, I was able to decouple the ‘vehicle dynamics (Open AI Gym)’ and ‘path following’ (Simulation script) cleanly

- I did not realize before that I could have ‘different’ vehicle states for Multicopter & Fixed-wing, but it certainly makes sense to decouple them.

END