03-27-2023 ---------------------------------------------------

Run initial modification of Matlab reinforcement learning example for flying robot

Modification includes adding -1 to the y-axis second derivative, akin to a constant acceleration “downwards” emulating gravity as well as saturating the action to [0, 1]

Results poor. No clear learning and agent fails to stabilize quadcopter

Actions probably should not be saturated since the actor network supplies output [-1, 1]. Solution for next attempt is biasing and scaling to [0, 1]

03-28-2023 ---------------------------------------------------

Results are promising, agent stabilizes quadcopter for “easy” initial conditions, 30 deg < theta < 150 deg (90 deg = vertical, in line with gravity), but not for hard cases.

Learning happens early but plateaus and no further progress made

Graphical user interface

Description automatically generated

Error: reward function penalizes non-zero theta -> rotated 90 deg away from vertical, the original flying robot tried to end pointing its thrusters in the +x direction, we want to point in +y so the thrusters can oppose gravity. Kind of surprising that the routine learned given this conflicting penalty value.

In next attempt add -90 deg to theta\_wrapped so distance from -90 is penalized.

03-29-2023 ---------------------------------------------------

Results quite good. Agent stabilizes quadcopter as before, but now can recover from more adverse initial conditions. Also note that the agent can drive to desired state (x=0,y=0,theta=90) from many IC’s if bounds on x, y, and time are removed. This probably will help training in future attempts.



Chart

Description automatically generated

Not sure what yellow means or why it’s so noisy…

Error: theta wraps to [-pi, pi], so reward function is biased penalizing rotations pointing int the [-x, -y] quadrant more.

In next attempt, wrap to pi and equalize contribution away from desired angle (90 deg)

Chart, line chart

Description automatically generated