The class code has been updated with a new release of FGI.py and FG_Data.txt (MinFltDat.xml is unchanged); be sure to use the new versions and not the original.

For this MP you will be in assigned teams. Following the posted class policies, each collaborative team should turn in a single assignment. The names and NetIDs of all members must be clearly given at the top. Each team member is responsible for every other team member fully benefiting from the exercise and fully understanding the joint solution.

You will write a PID controller class and use instances to control the pitch, roll, and air speed of the 172 using the elevator, aileron, and throttle respectively. Use the conventional form of PID with proportional, integral, and derivative coefficients kp, ki, kd. Implement the following:

```python
class PID:
    def __init__(self, kp, ki, kd, minU, maxU, initU=0.0):
        where minU, maxU, initU give the minimum, maximum, and initial output values.

    def reInit(self):
        This re-initializes the PID controller, forgetting any state information but preserving the target value.

    def newTarg(self, targetV):
        Calling newTarg sets the target or goal output value for the controlled system (e.g., the desired 172 roll angle).

    def ctl(self, curVal, curTm):
        Given the current output value of the controlled system (e.g., the roll angle) and the current time, it returns the input value for the controlled system (e.g., the aileron setting).
```

The PID controller is quite easy to write, but do not waste time or put off implementation. You must also find adequate kp, ki, kd parameter values. This can be tricky and frustrating. You may wish to implement some kind of helpful graphic interface to aid in experimentation (as demonstrated in class). You may wish to think about PID tuning in terms of the time periods $T_i$ and $T_d$ (termed the “standard form” in Wikipedia) but your implementation must use the conventional parameters kp, ki, kd. You may wish to follow one or another of the popular tuning methods, some of which are given in Wikipedia. It may be convenient to find approximate parameter values achieving somewhat controlled behavior and only then search in turn for the best parameter values. The best values achieve their desired targets quickly and from the widest initial state of the 172.

You will also need to implement code to launch FGI, to instantiate & run the PIDs, and to maintain communication among them. This code should enforce smooth control operation. For example, the PID may yield a desired elevator value which if set immediately and abruptly would damage the 172. Safely achieving the desired control settings is the responsibility of the communication code.

Your PID class must implement some solution to integrator windup so be sure you understand that phenomenon.

Hand in:

1) Your documented code

2) The best PID parameter values you can find for elevator, aileron, and throttle control of pitch, roll, and air speed.

3) A description of how you solved: a) enforcing smooth control changes, b) limiting PID output to valid Umin Umax, c) integrator windup.

4) A summary of your parameter tuning experience. Speculate on how this procedure might be automated with machine learning and what pitfalls you expect in doing so. Be sure to cite all important sources of information you use.

5) Your thoughts on the use of PID control to automate: a) landing and b) aerial dog fighting