WORKSHEET

U5.6 Saturation and windup

BugBotLab

Feedback control · University · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

What a controller should do when the actuator has nothing left to give.

- Where this robot saturates
- The symptom
- The fix
- Where else this bites

Questions

6 marks in all

1. A PI controller with conditional integration, run on four errors. What does it print? [1 mark]

```
KP, KI, DT = 1.5, 0.7, 0.1
integral = 0.0
for error in (30, 40, 30, 10):
    want = KP * error + KI * integral
    cmd = max(-55, min(55, want))
    if abs(want - cmd) < 0.01:
        integral += error * DT
    print(round(cmd, 2), round(integral, 2))
```

2. drive() is asked for 250 and then for 6. What does the robot get each time? [1 mark]

- ☐ **A** 100, then nothing
- ☐ **B** 250, then 6
- ☐ **C** 100, then 6
- ☐ **D** 100, then 15

3. What is the name of the anti-windup method that stops accumulating the integral while the actuator is saturated? [1 mark]

4. With back-calculation, integral -= (want - cmd) / Kt. If want = 62.1, cmd = 55 and Kt = 2, by how much is the integral reduced? Give two decimal places. [1 mark]

5. A robot switches from manual driving to a PI controller and jumps. What is the fix? [1 mark]

- ☐ **A** Initialise the integral so the controller's first output matches what was already being sent
- ☐ **B** Reset the integral to zero at the switch
- ☐ **C** Raise Kp during the switch
- ☐ **D** Clamp the command to 15 percent

6. A diagonal drive command saturates the lateral axis but not the forward one. What happens? [1 mark]

- ☐ **A** The robot travels in the wrong direction, because one component is clipped and the other is not
- ☐ **B** The robot slows down but keeps its direction
- ☐ **C** Nothing, because each axis is controlled separately
- ☐ **D** The robot stops

The task: anti-windup

The same blocker, the same 12 cm. Run a PI controller with anti-windup, keep `KI` at 0.7, plot `error` and `i term`, and be settled 12 cm from the blocker, within 3 cm, from 21 seconds onwards. Without protection the integral winds up on the way in and the robot is not settled until about 25 seconds; with it, about 15.

```
from bugbot import *
connect()

KP, KI = 1.5, 0.7
TARGET = 12.0
DT = 0.1
integral = 0.0
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/u5-6-saturation/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Run it with and without the anti-windup and put the two `i term` charts side by side.
2. Implement back-calculation instead and compare the recovery.
3. Add a lower limit: refuse to send a command below the dead band, and send zero instead. Does the loop behave better or worse?