ANSWERS

U5.6 Saturation and windup

BugBotLab

Feedback control · University · about 25 min

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))
```

Answer:

```
45.0 3.0
55 3.0
47.1 6.0
19.2 7.0
```

Tick 1 wants 45, not saturated, so the integral becomes 3. Tick 2 wants 62.1, clipped to 55, so the integral is frozen at 3. Ticks 3 and 4 are unsaturated and resume integrating to 6 and 7.

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

Answer: A. The drive clips at 100 at the top and does nothing below about 15 percent. Both are saturation, one at each end.

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

Answer: conditional integration. Conditional integration freezes the integral while the command is pinned, and resumes it the moment the controller has authority again.

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]

> **Answer:** 3.55 (accept within 0.005). (62.1 - 55) / 2 = 7.1 / 2 = 3.55. The integral is unwound in proportion to how much was clipped.

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

> **Answer:** A. That is bumpless transfer. Any system with modes needs it, because an integral left from before the switch produces a step in the command.

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

> **Answer:** A. The lateral axis is slower, so clipping it alone changes the ratio of the two components, and with it the direction.

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
```

The hint students can ask for: The command is pinned at its limit for most of the way in, and an unprotected integral keeps growing the whole time, so the robot arrives with far more push than it needs and drives into the blocker. Stop adding to the integral while the command is saturated, and it has much less to unwind.

A solution

```
from bugbot import *
connect()

KP, KI = 1.5, 0.7
TARGET = 12.0           # closer to the blocker than the robot can get
DT = 0.1
integral = 0.0

for tick in range(500):
    error = distance() - TARGET
    want = KP * error + KI * integral
    cmd = max(-55.0, min(55.0, want))
    # only add to the integral while the actuator still has something to give
    if abs(want - cmd) < 0.01:
        integral += error * DT
    plot("error", error)
    plot("i term", KI * integral)
    drive(cmd, 0, 0)
    wait(DT)
stop()
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.