ANSWERS

U10.6 How far behind

Following a trajectory · University · about 30 min

What this lesson is about

Dead band, lag and loop period, measured as the centimetres the robot is behind its own plan.

- Where the delay comes from
- The shape of the error
- Two ways to deal with it

Questions

7 marks in all

1. On the accelerating ramp the first order lag holds the velocity about a x tau below the demand. With a = 12 cm/s/s and tau = 0.25 s, how far below, in cm/s? [1 mark]

> **Answer:** 3 (accept within 0.01). 12 x 0.25 = 3 cm/s, which over a 1.2 s ramp builds a couple of centimetres of deficit.

2. The robot is 3 cm behind its reference while travelling at 15 cm/s. How far behind is that in seconds? [1 mark]

> **Answer:** 0.2 (accept within 0.001). A position error e at speed v is a time error e / v: 3 / 15 = 0.2 s, a fifth of a second.

3. Put the phases of the tracking error on a trapezoidal move in the order they happen. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `It settles once the reference stands still`

___ `It grows on the accelerating ramp`

___ `It goes negative on the decelerating ramp`

___ `It shrinks during the cruise`

> **Answer:**
>
> ```
> It grows on the accelerating ramp
> It shrinks during the cruise
> It goes negative on the decelerating ramp
> It settles once the reference stands still
> ```
>
> The error is built by the lag during the changes in the profile, not during the cruise, which is why a gentler acceleration shrinks it at both ends.

4. What does this print? It is the profile from the lesson. [1 mark]

```
D, A, V = 80.0, 12.0, 14.0
t_acc = V / A
d_acc = 0.5 * A * t_acc * t_acc
t_flat = (D - 2 * d_acc) / V
print(round(t_acc, 2), round(t_flat, 2), round(2 * t_acc + t_flat, 2))
```

Answer:

```
1.17 4.55 6.88
```

t_acc = 14 / 12 = 1.17 s and d_acc = 8.17 cm. The cruise covers 80 - 16.33 = 63.67 cm at 14 cm/s, 4.55 s, so the whole move is 6.88 s.

5. Why does pure pursuit degrade gracefully when the robot falls behind, and what does it give up? [1 mark]

☐ **A** It advances its target from the robot's own projection, not a clock, so the error cannot run away; but it has no opinion about timing

☐ **B** It uses a longer look-ahead when behind; but it cuts corners more

☐ **C** It feeds forward the reference speed; but it drifts without feedback

☐ **D** It slows the clock when the error grows; but it needs a trajectory

Answer: A. Pure pursuit is a path follower. The reference waits for the robot, and the price is that it will happily take all day.

6. Which of these add delay between the plan and the robot on the BugBot? [1 mark]

Tick every answer that is true.

☐ **A** The dead band, which makes the first few cm/s of the profile produce nothing

☐ **B** The 0.25 s first order lag in the drive

☐ **C** The 0.1 s loop period

☐ **D** Heavy filtering of the position being controlled on

☐ **E** Parameterising the path by arc length

Answer: A, B, C, D. All four delays add. Arc length is just how the path is described and costs no time.

7. A trajectory follower that slows its clock when the tracking error grows is using time scaling. What priority does that encode? [1 mark]

☐ **A** Geometry first, timing second, because being in the wrong place is worse than being late

☐ **B** Timing first, geometry second, because the plan promised a duration

☐ **C** Neither: it simply lowers the cruise speed for the whole route

☐ **D** It removes the need for feedback

Answer: A. Time scaling keeps the machine on the geometry when it cannot keep up with the feed rate.

The task: how far behind the plan

Build a trapezoid to 80 cm at 12 cm/s/s and 14 cm/s, follow it with feedforward and a correction, and keep holding the target for a few seconds afterwards. Plot `ref`, `actual` and `error`, and print `duration:`, what the profile says the move takes, and `worst:`, the furthest the robot ever fell behind the reference in centimetres.

```
from bugbot import *
connect()

DT = 0.1
DISTANCE, A_MAX, V_CRUISE, K = 80.0, 12.0, 14.0, 1.5
```

The hint students can ask for: Build a trapezoid to 80 cm at 12 cm/s/s and 14 cm/s, start a clock, and follow it with the reference speed fed forward and a correction on top. Keep the largest positive error you see, and hold the target for a few seconds afterwards so the run can be seen to have settled.

A solution

```
from bugbot import *
connect()

DT = 0.1
V_MAX = 20.0
DISTANCE, A_MAX, V_CRUISE, K = 80.0, 12.0, 14.0, 1.5

t_acc = V_CRUISE / A_MAX
d_acc = 0.5 * A_MAX * t_acc * t_acc
t_flat = (DISTANCE - 2 * d_acc) / V_CRUISE
duration = 2 * t_acc + t_flat
print("duration:", round(duration, 2))

def ref(t):
    """(position, speed) of the reference at time t"""
    if t <= 0:
        return (0.0, 0.0)
    if t < t_acc:
        return (0.5 * A_MAX * t * t, A_MAX * t)
    if t < t_acc + t_flat:
        return (d_acc + V_CRUISE * (t - t_acc), V_CRUISE)
    if t < duration:
        td = t - t_acc - t_flat
        return (d_acc + V_CRUISE * t_flat + V_CRUISE * td - 0.5 * A_MAX * td * td,
                V_CRUISE - A_MAX * td)
    return (DISTANCE, 0.0)

def band(u):
    if abs(u) < 5.0:
        return 0.0
    return max(17.0, min(100.0, u)) if u > 0 else min(-17.0, max(-100.0, u))

base = position()[1]
worst = 0.0
t0 = clock()
while clock() - t0 < 14.0:
    t = clock() - t0
    s, v = ref(t)
```

```
    here = position()[1] - base
    err = s - here                    # positive means the robot is behind the reference
    drive(band(100 * (v + K * err) / V_MAX), 0, 0)
    plot("ref", s)
    plot("actual", here)
    plot("error", err)
    if t < duration:
        worst = max(worst, err)
    wait(DT)
stop()
print("worst:", round(worst, 2))
```

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