WORKSHEET

U10.1 A path and a trajectory

BugBotLab

Following a trajectory · University · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

The same geometry with and without a clock attached, and why the clock changes what you can check.

- Arc length is the natural parameter
- The time law
- Why a moving target beats a fixed one
- Feasibility

Questions

6 marks in all

1. What is the practical reason for keeping the path and the time law separate? [1 mark]

- ☐ **A** The route can be re-timed for a low battery or a delicate payload without planning the geometry again
- ☐ **B** A trajectory cannot be checked for collisions, so the path must be kept for that
- ☐ **C** Arc length only exists for a path, not for a trajectory
- ☐ **D** The planner needs the time law before it can search the map

2. What does this print? It is the lesson's route with a constant speed time law. [1 mark]

```
import math
WAYPOINTS = [(40, 40), (40, 140), (140, 140), (140, 60)]
CRUISE = 12.0
length = 0.0
for (ax, ay), (bx, by) in zip(WAYPOINTS, WAYPOINTS[1:]):
    length += math.hypot(bx - ax, by - ay)
print(round(length, 1), round(length / CRUISE, 1))
```

3. Where does the reference say the robot should be 10 s into the route? [1 mark]

```
import math
WAYPOINTS = [(40, 40), (40, 140), (140, 140), (140, 60)]
CRUISE = 12.0
legs, length = [], 0.0
for (ax, ay), (bx, by) in zip(WAYPOINTS, WAYPOINTS[1:]):
    d = math.hypot(bx - ax, by - ay)
    legs.append((ax, ay, bx, by, d))
    length += d

def at(t):
    s = max(0.0, min(length, CRUISE * t))
    for ax, ay, bx, by, d in legs:
        if s <= d:
            u = s / d
            return ax + (bx - ax) * u, ay + (by - ay) * u
        s -= d
    return WAYPOINTS[-1]

print(at(10.0))
```

4. A robot drives at waypoint 1 until it is close, then at waypoint 2, and so on. Why does it stop and start at every waypoint? [1 mark]

- ☐ **A** Its demand is proportional to an error that collapses to nothing as it arrives at each waypoint
- ☐ **B** The planner inserted a pause at every waypoint
- ☐ **C** The drive cannot turn a corner while moving
- ☐ **D** The arc length is reset to zero at each waypoint

5. Which of these can be checked on a trajectory but not on the path alone? [1 mark]

Tick every answer that is true.

- ☐ **A** Whether any part demands more speed than the robot has
- ☐ **B** Whether it demands more acceleration than the drive can produce
- ☐ **C** How long the whole route will take
- ☐ **D** Whether the route passes through an obstacle
- ☐ **E** How long the route is in centimetres

6. Why does the lesson call the constant speed time law s(t) = v t a lie? [1 mark]

- ☐ **A** It jumps from rest to cruise speed instantly, which would need infinite acceleration
- ☐ **B** It gives the wrong total duration for the route
- ☐ **C** Arc length cannot be proportional to time
- ☐ **D** It only works on straight legs

The task: put a clock on a path

Print `length:`, the total length of the path through the four waypoints, and `duration:`, how long it takes at 12 cm/s. Then sample the trajectory every 0.1 s and plot `ref x` and `ref y`, with a `wait(0.1)` between samples so the chart spreads them out in time. The robot does not move in this task.

```
from bugbot import *
import math
connect()

WAYPOINTS = [(40, 40), (40, 140), (140, 140), (140, 60)]
CRUISE = 12.0
```

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

Do it on the robot

www.bugbotlab.com/learn/u10-1-path-and-trajectory/

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

Challenges

1. Re-time the same path at 6 cm/s. What changes, and what does not?
2. Plot the reference speed in x and in y. What happens at a corner, and why is that a problem for a real drive?
3. Add a fifth waypoint that doubles back on the path. Does your `at(t)` still behave?