ANSWERS

U2.4 Forward kinematics

Kinematics and frames · University · about 25 min

What this lesson is about

From the command to the body velocity to the path over the ground.

- Stage one: command to body velocity
- Stage two: body to world
- Stage three: integrate
- Predicting a run

Questions

6 marks in all

1. Put the stages of forward kinematics in order, from what you send to where the robot ends up. [1 mark]

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

___ `world velocity`
___ `position over time`
___ `body velocity, in cm/s`
___ `command, in percent`

Answer:

```
command, in percent
body velocity, in cm/s
world velocity
position over time
```

Scale by each axis's full scale, rotate by the current heading, then integrate.

2. The forward kinematics predict a 3 s run at forward(70) for a robot at heading 30, with V_MAX = 20. What does this print? [1 mark]

```
import math
V_MAX = 20.0
fwd, seconds, h_deg = 70, 3.0, 30
v = (fwd / 100) * V_MAX
h = math.radians(h_deg)
print("predicted x:", round(v * math.sin(h) * seconds, 1))
print("predicted y:", round(v * math.cos(h) * seconds, 1))
```

Answer:

```
predicted x: 21.0
predicted y: 36.4
```

v = 0.7 × 20 = 14 cm/s, so 42 cm in 3 s. Of that, 42 sin 30 = 21.0 goes along x and 42 cos 30 = 36.4 along y.

3. With W_MAX = 120 deg/s, how many degrees does drive(0, 0, 25) turn the robot in 3 s, by the forward kinematics? [1 mark]

Answer: 90. omega = 0.25 × 120 = 30 deg/s, and 30 × 3 = 90 degrees.

4. A forward kinematics prediction of a straight run is close but not exact. Which of these does the lesson say the model leaves out? [1 mark]

Tick every answer that is true.

- ☐ **A** The lag as the robot speeds up at the start
- ☐ **B** The coasting after the stop
- ☐ **C** The sideways leak from driving forward
- ☐ **D** A gain that is not quite what the data sheet says
- ☐ **E** The rotation matrix, which is only an approximation

> **Answer:** A, B, C, D. Those four are all real behaviour the simple model ignores. The rotation matrix is exact; what it is given is not.

5. A turning robot's position is predicted by x += wx * dt using the heading at the start of each step. When does the choice between that and the exact arc matter? [1 mark]

- ☐ **A** With a long step such as 0.5 s, because the heading changes a lot inside the step and the path is an arc, not a straight line
- ☐ **B** With a short step such as 0.02 s, because the errors add up over more steps
- ☐ **C** Only when the robot is driving straight
- ☐ **D** Never, because the heading is updated at the end of every step anyway

> **Answer:** A. Euler integration treats each step as a straight line at the starting heading. The smaller the step, the less the heading changes inside it and the smaller that error.

6. In stage two, which heading should rotate the body velocity into the world? [1 mark]

- ☐ **A** The heading the robot has now, updated every step
- ☐ **B** The heading at the start of the run
- ☐ **C** The heading the robot will have at the end of the run
- ☐ **D** The average of the starting heading and the target heading

> **Answer:** A. The body frame turns with the robot, so the rotation between body and world has to use the current heading each step.

The task: predict where it stops

Before driving, print where the forward kinematics say the robot will end up, as `predicted x:` and `predicted y:`. Then drive it and see. The robot does not start facing along an axis, so the rotation matters.

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

# predict first, then drive
```

The hint students can ask for: Work out the body velocity you are about to ask for, rotate it into the world with the heading you are at, multiply by how long you will drive, and print that before you drive. Then drive it.

A solution

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

v = 14.0           # roughly what 70 percent gives this drive
seconds = 3.0
h = math.radians(heading())
px = v * math.sin(h) * seconds
py = v * math.cos(h) * seconds
print("predicted x:", round(px, 1))
print("predicted y:", round(py, 1))

forward(70)
wait(seconds)
stop()
wait(0.6)
print("actually", position())
```

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