ANSWERS

U2.5 Inverse kinematics

Kinematics and frames · University · about 30 min

What this lesson is about

The useful direction: a velocity in the world, a heading, and the command that produces it.

- The problem
- In code
- Saturation

Questions

6 marks in all

1. The robot faces east, heading 90, and you want it to move due north at 12 cm/s. With V_MAX = 20 and V_LAT = 15, what does this print? [1 mark]

```
import math
V_MAX, V_LAT = 20.0, 15.0
h = math.radians(90)
wx, wy = 0.0, 12.0
vx = wx * math.cos(h) - wy * math.sin(h)
vy = wx * math.sin(h) + wy * math.cos(h)
print("fwd:", round(100 * vy / V_MAX), "lat:", round(100 * vx / V_LAT))
```

Answer:

```
fwd: 0 lat: -80
```

Rotated into the body, north is vx = -12, vy = 0: straight to the robot's left. -12 / 15 × 100 = -80 sideways and nothing forward.

2. The robot is at heading 180, facing -y. You want a world velocity of 12 cm/s due east (+x), with V_LAT = 15. What lat command does the inverse kinematics give? [1 mark]

Answer: -80. vx_body = 12 cos 180 - 0 = -12, so lat = 100 × -12 / 15 = -80. A robot facing towards you has its left hand pointing along +x.

3. In which order does the inverse kinematics work? [1 mark]

- ☐ **A** Rotate the world velocity into the body frame, then divide each component by that axis's full scale
- ☐ **B** Divide the world velocity by each axis's full scale, then rotate into the body frame
- ☐ **C** Rotate with body_to_world, then divide by the full scales
- ☐ **D** Divide wx by V_LAT and wy by V_MAX, with no rotation

Answer: A. The full scales belong to the robot's own axes, so the velocity has to be in the body frame before they apply. The rotation into the body is R(-h), the inverse of body_to_world.

4. The inverse kinematics asks for fwd = 150 and lat = 60. This scales the command down the way the lesson does. What does it print? [1 mark]

```
fwd, lat, rot = 150.0, 60.0, 0.0
worst = max(abs(fwd), abs(lat), abs(rot), 100.0)
fwd, lat, rot = fwd * 100 / worst, lat * 100 / worst, rot * 100 / worst
print(fwd, lat, rot)
```

Answer:

```
100.0 40.0 0.0
```

Every axis is divided by 150 / 100 = 1.5, so 150, 60, 0 becomes 100, 40, 0. The ratio between the axes, and so the direction, is kept.

5. The same request, fwd = 150 and lat = 60, is instead clipped axis by axis to 100 and 60. What does the robot do? [1 mark]

- ☐ **A** Travels in the wrong direction as well as too slowly
- ☐ **B** Travels in the right direction, just more slowly
- ☐ **C** Does exactly what was asked, because the robot clips internally anyway
- ☐ **D** Stops, because an out-of-range command is rejected

Answer: A. Clipping only the forward axis gets two thirds of it and all of the sideways, so the ratio changes from 150:60 to 100:60 and the path bends sideways.

6. Which request can a differential drive, with only a left and a right wheel speed, not satisfy? [1 mark]

- ☐ **A** A velocity straight sideways in its own frame
- ☐ **B** Driving forward while turning
- ☐ **C** Turning on the spot
- ☐ **D** Driving straight backwards

Answer: A. Two wheel speeds give forward speed and turn rate, nothing else. An arbitrary sideways request has no solution, which is exactly what non-holonomic means.

The task: north while facing east

The robot faces east. Get it into the green zone to the north without turning: heading must still be within 15 degrees of 90 at the end. Use the inverse kinematics.

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

V_MAX, V_LAT = 20.0, 15.0
```

The hint students can ask for: The robot faces east and must travel north without turning. Decide the world velocity you want, rotate it into the body frame with the current heading, and scale each axis by its own top speed.

A solution

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

V_MAX, V_LAT = 20.0, 15.0

def world_drive(wx, wy):
    """Ask for a velocity in the world, whatever way the robot is facing."""
    a = math.radians(heading())
    bx = wx * math.cos(a) - wy * math.sin(a)      # right, in the body frame
    by = wx * math.sin(a) + wy * math.cos(a)      # forward, in the body frame
    drive(100 * by / V_MAX, 100 * bx / V_LAT, 0)

while position()[1] < 95:
    world_drive(0, 12)
    wait(0.1)
stop()
```

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