ANSWERS

A3.7 Vectors

Data structures · A level · AQA 7517 4.2.1.4 · about 25 min

What this lesson is about

Vectors as lists, functions and arrows; addition, scaling, dot product and convex combination, on the robot's position.

- Three ways to see a vector
- Adding and scaling
- Convex combination
- Dot product
- BugBot's vectors

Questions

6 marks in all

1. What is the dot product of [2, -1, 3] and [4, 5, 1]? [1 mark]

> **Answer:** 6. 2 x 4 + (-1) x 5 + 3 x 1 = 8 - 5 + 3 = 6.

2. What does this program print? [1 mark]

```
u = [2, 5]
v = [6, 1]
w = [0.5 * u[i] + 0.5 * v[i] for i in range(2)]
print(w)
```

> **Answer:**
>
> ```
> [4.0, 3.0]
> ```
>
> With both weights 0.5 the convex combination is the midpoint of u and v.

3. For $\alpha u + \beta v$ to be a convex combination of u and v, which conditions must hold? [1 mark]

- ☐ **A** $\alpha \geq 0$, $\beta \geq 0$ and $\alpha + \beta = 1$
- ☐ **B** $\alpha + \beta = 0$
- ☐ **C** $\alpha = \beta$
- ☐ **D** $\alpha > 1$ and $\beta > 1$

> **Answer:** A. Those conditions put the result on the line segment between u and v.

4. The dot product of two non-zero vectors is 0. What does that tell you? [1 mark]

- ☐ **A** They are perpendicular
- ☐ **B** They point in the same direction
- ☐ **C** They have the same magnitude
- ☐ **D** One of them is the zero vector

> **Answer:** A. u . v = |u||v| cos(angle), and with non-zero vectors that is 0 only when cos(angle) = 0, at 90 degrees.

5. The vector [2.0, 3.5, -1.0] is a member of which set? [1 mark]

- ☐ **A** $\mathbb{R}^3$
- ☐ **B** $\mathbb{R}^2$
- ☐ **C** $\mathbb{N}^3$
- ☐ **D** $\mathbb{R}^1$

> **Answer:** A. It has three components, each a real number, so it is a 3-vector over $\mathbb{R}$.

6. What is the angle, in degrees, between the vectors [1, 0] and [1, 1]? [1 mark]

> **Answer:** 45. cos(angle) = 1 / (1 x √2) ≈ 0.707, and the angle whose cosine is 0.707 is 45 degrees.

The task: meet on the line

Two beacons are at `u = [40, 20]` and `v = [-20, 60]`, in cm from the robot's start (x to the right, y forward). Write these three functions yourself, for vectors of any length, without numpy: - `add(a, b)`: returns a new list, the vector sum of `a` and `b`. - `scale(k, a)`: returns a new list, the number `k` times the vector `a`. - `dot(a, b)`: returns the dot product of `a` and `b`, a single number. Then print, in this order: 1. `u.v = <dot product>`, which is a whole number. 2. `angle = <angle> degrees`, the angle between `u` and `v` rounded to a whole number with `round()`. You will need `math.acos`, `math.degrees` and `math.sqrt`. 3. `w = <w>`, where `w = 0.25u + 0.75v`, built with `scale` and `add`, printed as a Python list, for example `w = [1.0, 2.0]`. Finally drive to `w`: slide sideways by its x component (left if it is negative) and then forward by its y component.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
import math

u = [40, 20]
v = [-20, 60]
```

The hint students can ask for: The dot product multiplies matching components and adds the results. The angle comes from rearranging u.v = |u||v| cos(angle), and each length is the square root of a vector's dot product with itself. The convex combination is two scalings and an addition; its x component tells you how far to slide and which way.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
import math

u = [40, 20]
v = [-20, 60]

def add(a, b):
    return [a[i] + b[i] for i in range(len(a))]

def scale(k, a):
    return [k * x for x in a]

def dot(a, b):
    total = 0
    for i in range(len(a)):
        total = total + a[i] * b[i]
    return total

def length(a):
    return math.sqrt(dot(a, a))

print("u.v =", dot(u, v))
angle = math.degrees(math.acos(dot(u, v) / (length(u) * length(v))))
print("angle =", round(angle), "degrees")
w = add(scale(0.25, u), scale(0.75, v))
print("w =", w)

if w[0] > 0:
```

```
    right(50, distance=w[0])
else:
    left(50, distance=-w[0])
forward(50, distance=w[1])
```

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