ANSWERS

U3.8 Twists and the exponential

Odometry and drift · University · about 35 min

What this lesson is about

A steady command is a twist, a turn about one point. The exact odometry step, the matrix exponential behind it, and the logarithm that plans one smooth arc.

- A command held steady is a twist
- Every steady twist is a turn about one point
- The exact step
- How much it matters
- Going backwards: the logarithm

The task: land on the spot in one arc

The green square is 25 cm to the robot's right and 30 cm ahead of it. The robot must arrive in it facing 120 degrees (a third of a turn clockwise), using **one** steady `drive()` command held for `T = 3` seconds, then `stop()`. Work out the twist with the logarithm, print it as a line starting `twist:` with `w = ...`, `vx = ...` and `vy = ...` (`w` back in degrees a second, the speeds in cm/s), turn it into a command with the full-scale speeds below, and drive it. The printed twist is checked against the logarithm, to a tenth, and every number in it and in the `drive()` should be worked out by your program.

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

V_MAX, V_LAT, W_MAX = 20.0, 15.0, 120.0     # the design's full-scale speeds: cm/s forward,
cm/s sideways, deg/s
DX, DY, TURN = 25.0, 30.0, 120.0            # where to end up, in the start's body frame,
and how far to turn
T = 3.0                                     # how long to hold the command, s
```

The hint students can ask for: Work out w from the turn and the time, in radians a second, then C and S. The matrix line gives the end point from the twist: you have the end point and want the twist, so undo it (two equations, two unknowns; a 2 by 2 rotation-like matrix is easy to invert). Print the twist with w back in degrees. Divide each speed by its full scale to get a percentage, check none sits inside the 15 percent dead band or above 100, and hold that one drive() for exactly T before stopping.

A solution

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

V_MAX, V_LAT, W_MAX = 20.0, 15.0, 120.0
DX, DY, TURN = 25.0, 30.0, 120.0
T = 3.0

w = math.radians(TURN) / T                    # rad/s
C = math.sin(w * T) / w
S = (1 - math.cos(w * T)) / w
n = C * C + S * S
vx = (C * DX - S * DY) / n
vy = (S * DX + C * DY) / n
print("twist: w =", round(math.degrees(w), 2), "deg/s, vx =", round(vx, 2), "vy =",
round(vy, 2), "cm/s")

cmd = (100 * vy / V_MAX, 100 * vx / V_LAT, 100 * math.degrees(w) / W_MAX)
print("drive", [round(c, 1) for c in cmd])
drive(*cmd)
wait(T)
stop()
```

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