Assignment #3

In the provided Matlab source "obstacle_avoidance_learning.m", you are given a simulation (Fig. 1) of a Braitenberg robot (a green circle) located in an arena with a single chemical source (a black square). The chemical source has a Gaussian strength gradient, i.e. the magnitude of chemical concentration decreases in a Gaussian manner as the distance from the source increases. The robot (Fig. 2) has two chemical sensors, a distance sensor and two motors, each driving a wheel.

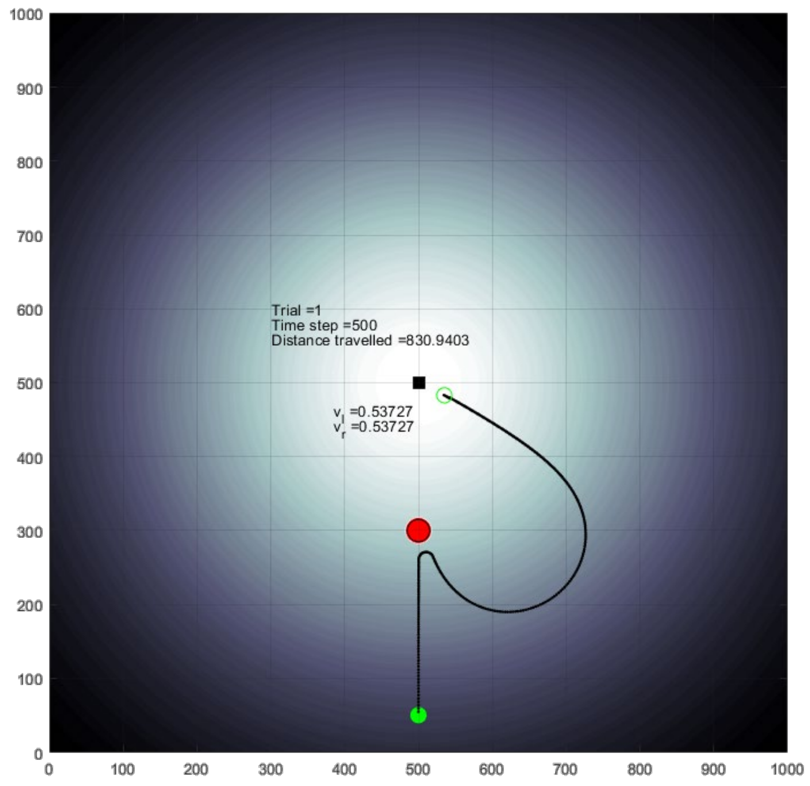

Fig. 1: Simulation environment

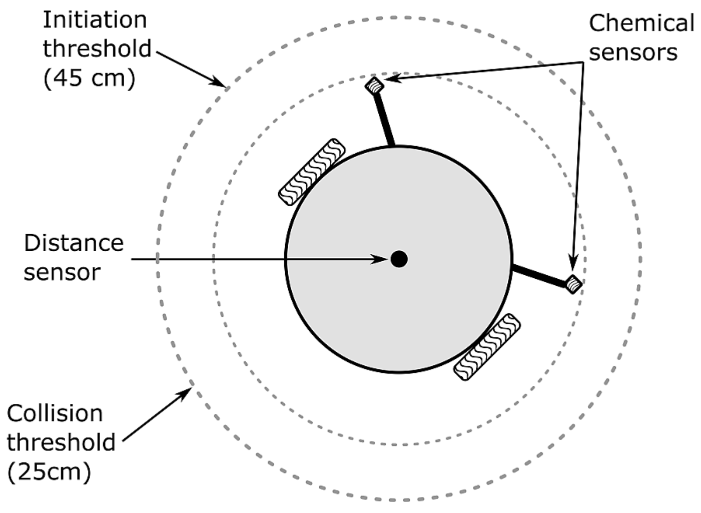

Fig. 2: Robot model

A bump sensor is simulated by using the distance sensor to detect an obstacle (red circle in Fig. 1) within the “collision threshold” (an imaginary barrier around the robot). The distance sensor can only detect an obstacle that is within the “initiation threshold” (another imaginary barrier around the robot) and return its distance from the robot. Both sensors have 3dB Gaussian noise.

The robot has two behaviours – (a) changing direction with fixed wheel velocities *v_l* (left) and *v_r* (right) when it “bumps” into an obstacle and (b) moving towards the chemical source using a Braitenberg model to convert chemical sensor signals into wheel velocities.

In each learning trial the robot moves from initial position to target. A single learning trial runs for max. 5000 timesteps (*ts*) and maximum learning trials are set to 50. A trial ends when either *ts* = 5000 or the robot reaches within 20 cm of the target or goes behind it. The simulation stops when the robot learns to avoid the obstacle without bumping into it and the final learned trajectory is highlighted in green or all 50 learning trials are used up.

1. Implement Input Correlation (ICO) learning algorithm to learn how to avoid the obstacle in the environment. Following sensor signals are available to learn obstacle avoidance –

   bump_sensor(ts) = bump sensor signal at timestep *ts*

   (0 = no bump, greater than 0 = bump detected)

   distance_sensor(ts) = distance sensor signal at timestep *ts*

   Essentially, you must implement two equations –

   (a) Synaptic weight update:

   $$w_{new} = w_{old} + \mu \cdot predictive\ signal \cdot \frac{d(reflex\ signal)}{dt}$$

   μ = learning rate (*mu* in the code, default value = 0.2)

   w = synaptic weight (*w* in the code, default initial value = 0)

   You may calculate derivative $\frac{d}{dt}$ over as many time steps as you like (minimum=1).

   (b) Motor update:

   $$v = w \cdot\ predictive\ signal + 1 \cdot reflex\ signal$$

   Here the weight of reflex signal set to 1, as we do not update this in ICO learning.

   The motor update v is automatically added to either motor velocity *v_l* or *v_r* depending on whether the robot needs to respectively turn right or left when avoiding obstacles.

   Add your code for these equations between the following two lines in the code:

   ```
   %%%%%%%%%%%%%%%%%% Add your code here %%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ```

Run the code and observe the effect of learning on the trajectories of the robot (black lines as in Fig. 1) in each trial. Try to think about why the robot's trajectory changes, from the perspective of equations (a) and (b).

2. Repeat question 1 for learning rate *mu* set to 0.3 and then to 0.4. Think about what effect increasing learning rate has as well as why, from the perspective of equations (a) and (b), on the number of trials taken to learn obstacle avoidance.

3. Add environmental noise to the simulation by setting variable *add_noise* = 1 and repeat questions 1 and 2. Think about what effect adding environmental noise has on the trajectories as well as on the number of trials taken to learn obstacle avoidance. Is it different than when there is no environmental noise? If so, why do you think that is?

Hint #1: For counting number of trials, you may use the plot on the bottom right (an example is shown in Fig. 3 below) where the y-axis is the distance travelled by the robot in each trial. The end of every trial is marked by a blue circle. This plot is generated every time the code is run.

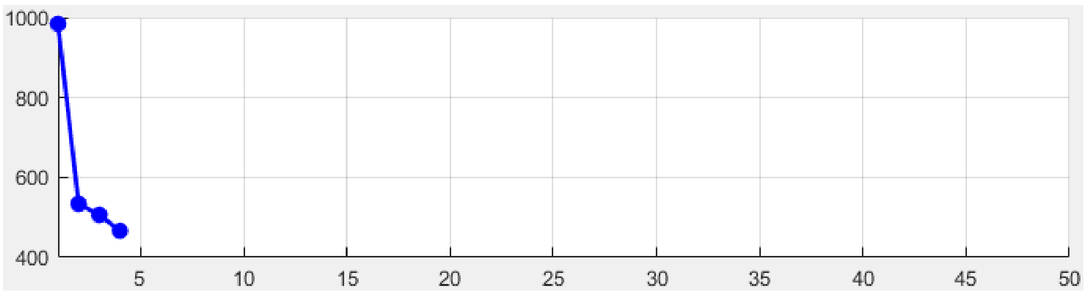

Fig. 3: Example plot showing distance travelled in each learning trial.

Hint #2: To see if the robot is really learning to avoiding bumping into the obstacle, set the parameter *subfigs_on* to 1 before running the simulation. This will generate a plot similar to that shown in Fig. 4 below, where the green line indicates whether the bump sensor is triggered or not. No spikes in this line indicate that the robot has not bumped into the obstacle while spikes indicate that the robot has bumped into the obstacle.

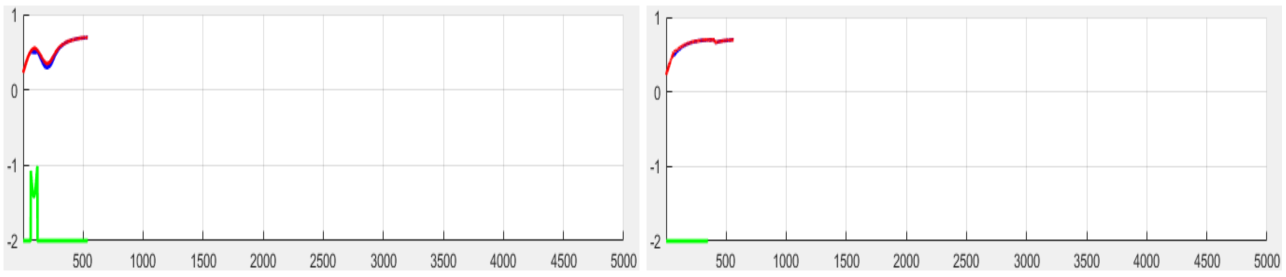

Fig. 4: Sensor data during learning (left plot) and after learning (right plot). Red (left sensor)and blue (right sensor) lines represent chemical sensor data, while the green line indicates bump sensor data (shifted artificially downwards for better visibility).