Task Sheet 5 - Collective decision-making: urn model, global switching, foraging

Objectives

A. Implement an empirical approach to the application of the urn model

B. Test a simple model for how accurately it represents reality

C. Implement and investigate foraging and measure swarm performance

1 Urn model for locust scenario

We use the locust simulation again. Reuse your code from the last tutorial (tutorial 4, task 1: Dimension reduction and modeling) or implement it now. This week we work with the following parameters: circumference $C = 0.5$, speed of 0.01, perception range of $r = 0.045$, swarm size of $N = 50$, spontaneous direction switch with a probability $P = 0.15$ per time step.

In your simulation, you still have to keep track of the left-goers L_t at a time step t and count them. This week we are interested in the average change of L_t in dependence on itself. That is, we are searching for a function $\Delta L(L)$. To measure this function, we simulate the locusts for 100 time steps. These 100 time steps are needed by the locusts to organize themselves. We simulate an additional 20 time steps while we keep track of L_t. At each time step $t \in [101, 120]$ we 'store' the change of L within one time step $\Delta L = L_t - L_{t-1}$, for example, by adding it to an array. It's an array because we want to measure ΔL in dependence on L itself. Hence, you can have a variable *deltaSum*[] in your program and for each time step $t \in [101, 120]$ you do: $deltaSum[L_{t-1}] += L_t - L_{t-1}$. Also, keep track of how often you have added something to an entry (e.g., a variable *count*[]) to normalize your measurements. Repeat these runs many times – up to 50,000 samples could be useful (fast code will pay off). Finally, write your data of the measured function $\Delta L(L)$ to a file for later use.

A. Plot the data for $\Delta L(L)$. What is the mathematical and domain-specific meaning of the positions L^* with $\Delta L(L^*) = 0$?

B. Next, we fit the swarm urn model to this data. Remember the equations

$$P_{FB}(s, \varphi) = \varphi \sin(\pi s), \tag{1}$$

$$\Delta s(s) = 4\left(P_{FB}(s, \varphi) - \frac{1}{2}\right)\left(s - \frac{1}{2}\right), \tag{2}$$

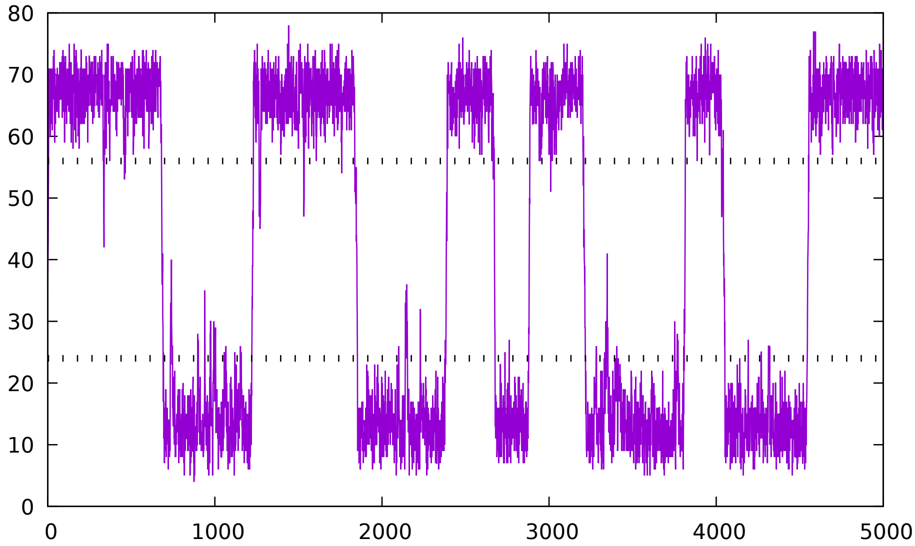
whereas P_{FB} is the probability of positive feedback and $\Delta s(s)$ is the expected average change of the considered state variable. Here, these are the left-goers, and hence s is the ratio of left-goers $s = L/N$. In addition, we have to introduce a scaling constant c to the idealized equation, which yields:

$$\Delta s(s) = 4c\left(P_{FB}(s,\varphi) - \frac{1}{2}\right)\left(s - \frac{1}{2}\right) \tag{3}$$

Fit this function to your data, that is, try to find appropriate values for the scaling constant c and the feedback intensity φ. Be sure to keep φ within a reasonable interval: $\varphi \in [0, 1]$.

C. Plot your data together with the fitted function. Also, plot the resulting probability of positive feedback P_{FB}. What can be said about these results?

2 Density-dependent global switching



We use the locust simulation once more. In the biological system with actual locusts, a density-dependent global switching behavior was observed. Even though the swarm had converged to either a majority of left-goers or right-goers, it switched sometimes to the other majority. For low densities, this was observed more frequently than for high densities. Here we test whether we see the same in our little simulation.

We use the parameter set as in task 1 except for the swarm size N that we want to vary. We test a number of swarm sizes on the interval $N \in [20, 150]$ (fast code will pay off). We measure the times between observed global switches in the following way. For a given swarm size N, we Define three 'zones' depending on the observed left-goers L:

Zone A: $L > 0.7N$

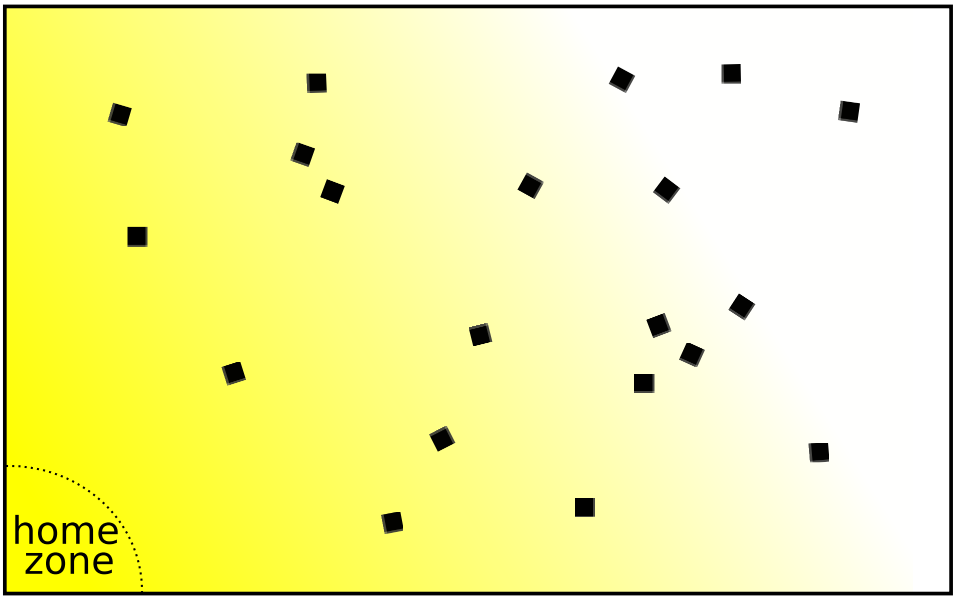
Zone B: $0.3N \leq L \leq 0.7N$

Zone C: $L < 0.3N$

We measure the times it takes the swarm to go either from zone A to zone C or from zone C to zone A. Whenever the swarm is in zone A or zone C we set a counter to zero. Once we enter zone B we start to count and remember from which zone we came from. When we then leave zone B into the other zone (i.e., the zone we did not come from), we store the time on the counter, set it to zero, and continue. If we leave zone B into the zone we came from we reset the counter to zero without storing the time.

This way we can collect durations of time between two 'global switches.' Measure these durations for a number of swarm sizes $N \in [20, 150]$ for several independent simulation runs each, with each running for at least 5,000 time steps, but better more. Make sure you observe enough switches for each tested swarm size. Calculate the average observed global switch time for each tested swarm size and plot them over swarm size. Also plot the number of observed switches over swarm size. Do we observe something similar as in the natural swarm?

3 Forging



In this task we go back to our collective robotics simulation. Implement a robot controller for foraging that satisfies the following specifications:

A. The robot is supposed to collect objects and take them to a home zone marked with bright light.

B. The robot has a number of proximity sensors, and in addition, a special kind of bumper sensor to detect when it is pushing against objects. A force is measured that allows to tell how many objectes its pushing (assuming all objects have the same size, weight, and friction etc.).

C. The robot has also two light sensors (left and right).

D. Your controller code that is executed in each cycle should output the following values:

   (a) actuator value for motor left/right
   (b) error (false, true, or unknown)
   (c) collision (false, true, or unknown)
   (d) arena boundary (false, true, or unknown)
   (e) transporting object (false, true, or unknown)
   (f) home zone (false, true, or unknown)

Run the experiment with different swarm sizes $N \in \{1, 2, \ldots, 10\}$ (or even more). Measure the swarm performance (collected objects during a fixed period of time). Plot the swarm performance over swarm size.

Your Submissions

- Please zip your submission in a single file
- Include the source code and a detailed readme
- For tasks 1, 2, and 3, provide written solutions and plots.