M20 - Introduction to Computer Programming with MATLAB
Instructor: Prof. Enrique López Droguett, Ph.D.
Teacher Assistants: M. Fidansoy, M. Pishahang, G. San Martín.
Summer 2023 - UCLA

FINAL PROJECT: ANT COLONY SIMULATION

Important information:

- This is a project that needs to be solved in groups of two. You will need to fill out a Google Forms specifying your partner (one answer per group is enough) **before July 31, 23:59**. Everyone that has not answered this form by that date will be assigned a random classmate.
- As you will see, this project will contain many sources of randomness. To make the executions replicable on your end and our end, we will use "random seeds". We will explain this concept during the discussions, but keep in mind that you will be assigned a line of code that you will have to use at the beginning of your code.
- **The deadline for the project is August 25th, 23:59. No extensions will be granted.**
- The submission will be through BruinLearn. Each group will have to submit **one** project package including:
    - PDF report detailing your solution and explaining how you optimized the simulation parameters.
    - Zip file with all the MATLAB code required to replicate your results and simulations.
    - Zip file with the videos generated in your simulations.
- We highly encourage you to start working **early**. No extra discussion sessions, or office hours will be granted for the project, **so do not wait until the last week to start asking questions.** This is not a project you can start and finish in a couple of days.
- A fixed code structure will be provided for you to help you in the development of the solution. While you are allowed to develop extra functions if required, **it is mandatory to adhere to the structure we will provide during the discussion session.**

Ant colonies are an excellent example of natural systems. They are composed by millions of small entities (the ants) that follow a surprisingly simple set of rules and are capable of interact, communicate, and fulfill multiple purposes with higher levels of efficiency. In this project, we will explore the phenomenon of food recollection in an ant colony using MATLAB.

For this, your task is to implement in code the simple algorithmic behavior of an ant, and then extrapolate that to multiple individuals to see how the colony behaves in a simple 2D environment. In what follows, this document outlines the description of the ant behavior, the main requirements you need to fulfill and the way in which you should present your results. However, it is highly encouraged that you assist or watch the discussion sessions, as we will go over different key suggestions that will help you solve the most challenging aspects of the project .

First, it is necessary to explain the environment in which our simulation will be developed. We will assume a rectangular 2D region described by two cartesian points. These points characterize the lower left corner and upper right corner of the considered region. This region is continuous, meaning that we will not discretize space: the ant will be able to move to whatever direction it wants in this region. However, the simulation will be discretized in its time domain; we will compute the position and states of the ants at intervals separated by Δt seconds.

This 2D map will contain the colony, located at (x_{colony}, y_{colony}), and the position of multiple food elements. The food locations will be modeled as a matrix, where each row will represent a different food location, and the columns will be the x and y positions of the food element. For this project, we will provide you with 2 maps of different size and complexity. You can consider these maps as an input for your simulation. Each map will consist of:

- Two cartesian points describing the lower left and upper right corners of the region: (x_1, y_1) and (x_2, y_2).
- The colony position and radius: (x_{colony}, y_{colony}) and r_{colony}, respectively.
- A matrix F listing the food sources.

Figure 1 shows a MATLAB plot depicting a simplified environment.

Now, we will explain how the ants should be modelled. For this project, each ant will have 4 attributes that you will need to keep track of and update as required. These attributes are:

- The x and y positions of each ant in the map.
- The facing angle ϕ with respect to the horizontal x-axis. You can think of this parameter as the direction that the ant is currently facing (or "looking").
- A Boolean parameter that indicates whether the ant is currently carrying food or not. This is important because ants act very differently depending on their carrying status.

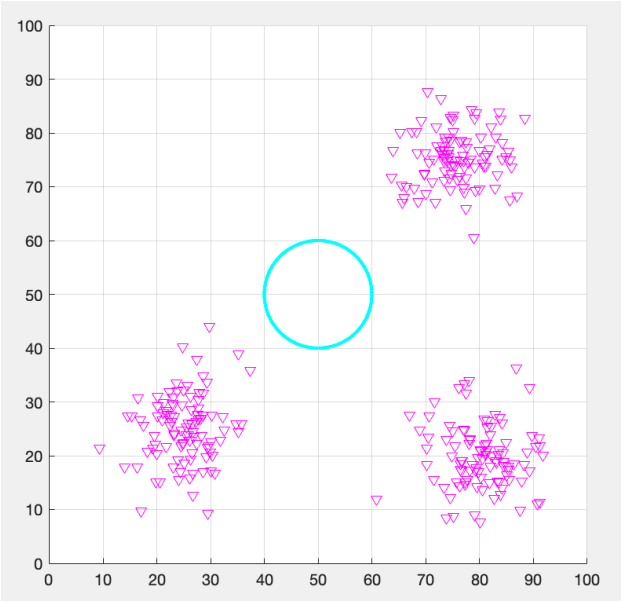

Figure 1: Map composed by the region described by $(x_1, y_1) = (0,0)$ *and* $(x_2, y_2) = (100,100)$. *Magenta triangles are the food sources. The cyan circle is the colony. In this case,* $(x_{colony}, y_{colony}) = (50,50)$ *and* $r_{colony} = 10$.

Before delving into the actual ant model, there is a core concept that needs to be explained. The way ants communicate with each other is through special chemical molecules called pheromones. While real ants have multiple pheromones for multiple purposes, we will only consider two in this project: the blue and the red pheromone. The blue pheromone is dropped by ants when they are looking for food, as a way of guiding them back to the colony once they find the food source. You can think of this pheromone as the "trail back home". The red pheromone is dropped by the ants when they have found food, to show other ants where to locate food sources. You can think of this pheromone as the "way to the food".

With this in mind, let us explain the behavioral model for ants that you will have to code in this project. We will assume that **each** ant, at **each** time step in the simulation perform the following tasks:

1. The ant will "smell" an area in front of it, as indicated in Figure 2. Then, it will turn towards the point with the higher concentration of pheromones plus a random angle controlled by $N(0, \sigma_{\phi_1})$. This point is computed as the average point of all the pheromones positions in the area weighted by their respective concentrations. If there are no pheromones in the area, the ant will shift its angle in $N(0, \sigma_{\phi_2})$ radians. This random shift will add some variability to the ant movement, so it will favor the exploration of the map. Remember that ants without food will only "smell" for the red pheromone, while ants with food will "smell" the blue pheromone instead.

You are free to tune the parameters σ_{ϕ_1} and σ_{ϕ_2} as you see fit. These parameters will control the trade back between exploration of new food sources and exploitation of known food sources.

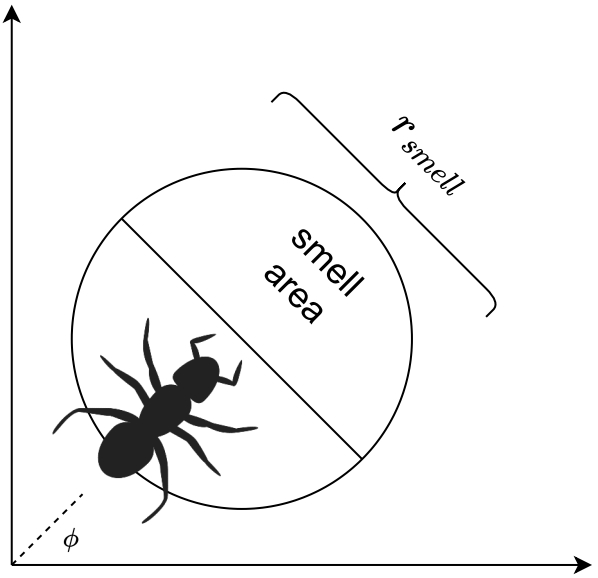

Figure 2: The ant is positioned with a facing angle ϕ. r_{smell} is a parameter that you can tune within a range specified at the end of this document.

2. An ant will try to move **one** unit of distance in the updated angle the ant is facing. If the move causes the ant to be off-limits (out of the map), then the move is cancelled and the ant's facing direction is shifted in π radians (it will turn 180 degrees). However, if the move does not violate any boundaries in the map, then the move is valid and therefore executed, updating the ant's position in the map.
3. Depending on their food status, the ants will drop the corresponding pheromone (remember that ants with food drop the red pheromone, ants without food drop blue pheromone). The pheromone will be dropped at the ant's position. You will need to keep track of the pheromones' position, as they are used in step 1 by ants to decide where to go.
4. Finally, if an ant is not carrying food **and** is close to one or multiple food sources by less than r_{food} units of distance, then the ant will pick up the closest food source. The food source is consumed (and therefore eliminated from the matrix *F*), and the ant shifts its facing angle in π radians to start travelling back to the colony. If the ant is carrying food **and** the colony center is within a radius of r_{colony}, then the ant drops the food in the colony and the ant shifts its facing angle in π radians to start looking for food again. In this case, the colony increases its food count by one. **You need to keep track of how much food is collected, as it will be the main measure of how efficient your colony is.**

Now, let us explain the pheromone model that you will have to implement. Both blue and red pheromones have a limited duration in the environment. This is a fundamental aspects of ant colonies, as it allows ants to "forget" paths that lead to exhausted food sources. This phenomenon can be implemented in multiple ways, but for this project we will follow a simple rule: linear decay. We will assume that **each** pheromone is dropped with an initial concentration equal to one. Then, during **each** timestep, the pheromone will decrease its concentration by an amount given by δ_{red} or δ_{blue}, depending on its type. At the end of each time-step, all pheromones that have reached a concentration lower than 0 will disappear from the simulation. **In Figure 2, the concentration for each sensing area will be computed as the sum of the concentrations of all the pheromones inside the area.** Additionally, we will assume that food smells like the red pheromone with an extremely high (infinite) concentration. On the other hand, the colony smells like the blue pheromone with an extremely concentration (infinite) as well. This will help our ants in their foraging mission.

Note that there are multiple parameters that we have been using through this explanation. Some of these parameters will be fixed and distributed along with the map descriptions, while for others you will have the opportunity to tune them within a range so that your colony maximizes the amount of food collected in a certain amount of time steps T. The following is a list of the parameters you can tune, including their allowed range:

- $\delta_{blue} \in (0, 1)$
- $\delta_{red} \in (0, 1)$
- $r_{smell} = [0,\ 10]$
- $\sigma_{\phi_1} = [0, +\infty]$
- $\sigma_{\phi_2} = [0, +\infty]$

Every other parameter is fixed for each simulation and will be provided along with the map descriptions.

To finish the modeling aspect of the project, let us discuss the initialization of the simulation. For each map, N ants will start at the center of the colony with an initial facing angle for each ant will be obtained from a uniform distribution between 0 and 2π.

Now, let us review the visualization aspect of this project. While strictly speaking you can run a simulation and obtain a final food collection result without visualizing anything, that is not the idea for this project. As such, an important requirement is to create a couple of visualization videos depicting your best performing colonies for each map (we will discuss what constitutes a performant colony in the next paragraph) . For this, you will record a

video where one frame will be equal to one timestep of the simulation. The video will be a 2D plot with the following characteristics:

- Ants are depicted as black asterisks
- Blue and red pheromones are depicted as blue and red dots with an opacity controlled by their concentration.
- Food sources are depicted as magenta triangles.
- The colony is depicted as a yellow circle of radius r_{colony}.

We will review examples of this plots and give some hints about the animation during the discussion sessions. Additionally, we will upload a video to BruinLearn showing what we will be expecting of your submission package.

Project Requirements

To get full marks in this project, you need to fulfill the following task.

- Create the required code that accepts as inputs the map parameters and run a simulation for the specified T time-steps (each map will have a different time-step value). Your code should keep track of the amount of food elements collected by the colony. Both the current time step and the amount of food collected should be displayed in the title of the animation (for an example, refer to the video that we will upload to BruinLearn). The code should be generic, meaning that it must accept generic map parameters. **You cannot hardcode the three map instances that we will provide for you.**
- Create the 2 videos (3 if you are solving the extra credit problem) showing the results of your simulation. The required format of the video is .mp4.
- Write a short report (5 pages of content) clearly stating the problem to solve, the way you organized and implemented your solution, and the strategy you followed to maximize the amount of food your colonies collected. While no extra points will be awarded for the final performances your colonies achieve, to get full marks you need to show in the report that you understand what each parameter is doing, and therefore you need to give a sensible explanation of your chosen values. You can use as a base the report that we uploaded for the assignments.

Extra-Credit (20 points):

The environments we have considered for these simulations do not include any type of internal boundaries. But this is not the case in nature. For this extra-credit part, you will modify your code to consider walls. Walls will be modelled as rectangles in the map. These rectangles will be always wider and taller than one unit of distance. When an ant collides

with a wall, the movement is considered invalid (and therefore is not executed, like what happens with the borders of the map), and it shifts its angle in π radians. An extra map considering walls will be provided to you. The walls in this map will be encoded as pairs of two points, describing the lower left corner and the upper right corner of each rectangle. Figure 3 shows a simple example of a map including two walls. To get the full 20 points, your code must be generic, i.e., it has to accept maps with an arbitrary number of walls. You **cannot** hardcode the example we will give you.

You need to generate an extra video for this map. In this video, walls need to be visualized as rectangles filled with the color black. Allocate an extra page of content in your report to explain how you implemented the walls into the simulation and what changes did you need to do to the parameters in order to allow for an efficient recollection.

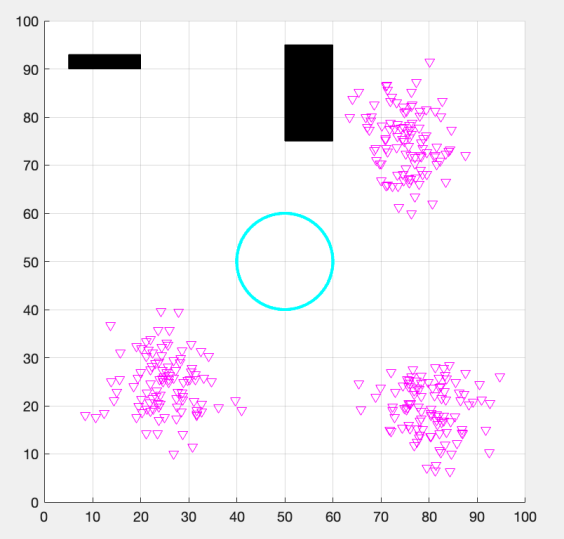

Figure 3: Map with two walls.