**Questions regarding the current model**

1. Evaluate whether the pheromone decline and detection is not too strict:
   * Currently, pheromone levels decline in halves to adjacent cells = **leave\_pheromone()**
   * The minimal level to be detectable by an ant is set at 20 (= **pheromone\_min\_detectable**)
   * The evaporation as the ant goes back to the nest is exponential = **leave\_pheromones()**
   * The pheromone evaporates naturally by an evaporation rate currently set to 0.05 (so 5% decline/step) = **update\_pheromones()**
2. When the ant returns, the level of pheromone it deposits will exponentially decrease as it gets closer to the colony. This solves the issue of having to go up against the gradient; because the level at the food source will be higher than the level at the colony, thus the ants can simply follow the highest value of pheromone when searching for food.
3. When an ant returns and encounters a searching ant, the searching ant should not follow the returning ant but keep on searching

🡪what happens in biology?

🡪currently, the ant deposits exponentially less pheromone when approaching the colony, so the searching ant will eventually go back up to the higher values?

Two main solutions to the problem (if we want to address it):

* Include ant-to-ant interactions 🡪 implement code that allows ants to be in the same cell and lets them detect the presence of other ants
* Implement directional pheromones (searching and returning pheromone)

1. Now, because it seemed that way in the introduction file by Kamiel, we only allow the ants to move in the cardinal directions (right, left, up, down). We could also implement that they can move along diagonals. The pheromones do diffuse along diagonals too (so along all directions).
2. Currently, there is no else statement in the **random\_walk()** function (line 105). This means that the ant stays in the same position if the new position is outside the bounds of the arena. Essentially, this means that when the ant bumps into the ‘walls’ of the arena, it stands still for a step and realises there is an obstacle, after which it will continue to move. Biologically this kind of makes sense? We can also implement an else statement that indicates that it should calculate a new position if the position is out of bounds.

🡪change it to else statement

**Next steps**

* Run the model in different scenarios:
  + Without pheromones
  + With pheromones 🡪 vary the number and levels of the food sources
* Decide on the parameters:
  + How many runs, how many steps/ant before end of the run?
  + How many ants at the same time?
  + What formula for the evaporation rate? How quickly does the deposited pheromone decline in terms of distance from food source? What diffusion rate? What level is the minimal detectable threshold?
* Grid size? Colony size (determines nr of random exits from the colony)?
* Make the figures:
  + Heat map
  + Animation
  + Comparison between the different conditions (ggplot lines?)
* Conduct the statistics on the obtained datasets => also still implement a function that actually stores the dataset somewhere!

**After adaptations**

* We assume that the ant always follows the highest pheromone value 🡪 we could add probabilities to this?
  + Let ants probabilistically follow stronger pheromone trails but still have a chance to explore less-marked paths
* Should add diffusion over time? 🡪 now the diffusion occurs once (at deposition by ant) to adjacent cells (also diagonally) by 50% and not to the cells neighbouring cells (25%)
* Option 1: as ant continue to move, the pheromones diffuse further away from the trail. Eventually they are replaced with 0 when they go below the threshold
* Option 2: when the ant deposits, the pheromone diffuses immediately further away. As the ant continues to move, the pheromone doesn’t diffuse anymore.
* Cardinal directions or include diagonals as well
* Line 66 (update\_pheromones):

# Ensure that there are no negative values

self.pheromone\_grid = np.maximum(self.pheromone\_grid, 0)

# Ensures that there are no negative values in the grid, replaces those with 0 (level is positive / 0 = remains unchanged <=> level is negatiev = replaced with 0)

* In the current implementation, the line ensuring no negative values may seem redundant because the thresholding step also sets values below pheromone\_min\_detectable (which includes negative values) to zero
* Due to the nature of floating-point arithmetic in computers, operations that should mathematically yield non-negative results might sometimes produce tiny negative values (e.g., -1e-16).

**Example:** If a pheromone value is extremely close to zero and you perform arithmetic operations on it, the result might be a very small negative number due to rounding errors.

* Including a step to ensure no negative values is a form of defensive programming, adding robustness to your code by guarding against unexpected states
* Diffusion:

Current code: pheromone level in a given cell is only updated through diffusion if the current level of pheromone is lower

current cell = 25 adjacent cell = 20 🡪 a value of 10 would diffuse to the current cell

=>no update, current cell remains at 25

current cell = 25, adjacent cell = 60 🡪 a value of 30 would diffuse to the current cell

=>update: current cell becomes 30

Modify to?: pheromone levels are additive so in scenario 1 the current level would be changed to 35, in the 2nd one to 55

Potential problem: decay goes too slowly since pheromone levels are being added the whole time

**Side note follow\_pheromones():**

* If adjacent cells have different pheromone levels 🡪 ant moves to the cell with the highest pheromone level
* If adjacent cells all have the same pheromone level 🡪 ant randomly chooses one of these cells (basically local random walk)
* If all adjacent cells have zero pheromone 🡪 ant performs a random walk

**Summary pheromone behavior in current model (model\_try2):**

1. **Pheromone deposition:**

* When pheromones are deposited**:**
  + Ants deposit pheromones along their path when they return to the colony after finding food
  + The leave\_pheromone() method in the Ant class handles pheromone deposition
* **Pheromone value calculation:** pheromone strength decreases exponentially with each step away from the food source.

1. **Pheromone diffusion**:

* Immediate diffusion upon deposition**:**
  + The pheromone\_diffusion() method recursively diffuses pheromones to neighboring cells
  + Mechanism:
    - Start: pheromone is first deposited at the ant's current position.
    - Diffusion**:** pheromone value at each cell is halved and then spread to all adjacent cells (including diagonals)
    - Stop: diffusion continues until the pheromone value drops below the pheromone\_min\_detectable threshold (e.g., 25).
* Diffusion mechanics**:** the cell receives the maximum of its existing pheromone value and the new value being diffused
* Example of Diffusion:
  + Initial cell receives the full pheromone value (e.g., 100)
  + First layer neighbors receive half the value (e.g., 50)
  + Second layer neighbors teceive half of the previous value (e.g., 25)
  + Continued diffusion stops when the halved value is below 25

1. **Pheromone evaporation over time:**

* Evaporation mechanism**:** update\_pheromones() method in the Arena class.
  + At each simulation step, all pheromone values in the grid are reduced by a fixed evaporation rate.
* Thresholding: after evaporation, any pheromone values below the pheromone\_min\_detectable threshold are set to zero

1. **Ant movement influenced by pheromones**

* When searching for food, ants check for pheromones in adjacent cells
* Ants move to the neighboring cell with the highest pheromone concentration.
* If multiple cells have the same highest level, the ant chooses randomly among them.
* If all adjacent cells have pheromone levels below the detectable threshold, the ant performs a random walk.