FLIGHT OF THE BUMBLEBEES: AN ATTRACTION-BASED SIMULATION

A. DE LA BRUERE, J. GERTSCH, S. LAYTON, M. MELLA, W. STODDARD

Abstract. We model the motion of agents navigating towards a target destination in the face of distracting attractors. Specifically, in this paper we model the flight of bumblebees returning to their hive through a field of flowers. Our resulting model is a first-order dynamical system of independent agents as they face distractions against a primary objective. This research fits into a larger body of work focused on the study of animal movement and migration, typically modeled using stochastic or partial differential equations. Our research is unique in its simulation and treatment of distractions along the path of motion. The implications of this work are not only relevant to animal movement, but also contain potential applications to behavioral economic models.

1. Background/Motivation

The goal of this model is to explain the behavior of free agents progressing toward a destination while navigating through a field of various distractions. We chose to model the flight of bees on their way back to a hive (the destination) through a field of flowers (the distractions). We began deriving our model using principles from gravity [Hum23], where attraction to any destination or distraction was influenced by both proximity (distance) and preferences (a proxy for mass).

Originally, we considered this simulating this problem in the context of customers in a grocery store. Customers were tasked with locating a specific item while facing distractions such as ads, sales, and product samples. Similarly, this type of model could represent tourists in a city navigating to a famous landmark along a street full of tourist traps. However, a fixed store or city layout was more restrictive than we cared for. In order to give sufficient attention to the movement and behavior of an agent faced with various distractions, we abstracted this idea to bees in an unencumbered field. Studying the model we created can give intuition on the influence of proximity and preference on the motion of an agent through space that can be generalized to more specific settings and spaces.

Groups such as grocery retailers and city planners have incentive to care about modeling the movement of agents through space. Retailers understanding this problem could arrange a store optimally to make shopping trips more efficient and also to maximize revenue with clever placement of

goods. City planners could optimize traffic flow and the construction of new buildings to account for hotspots in their city.

Modeling the motion of animals has merit in its own right. There is a large body of research focused on animal movement and migration, published in journals focused on ecology, physics, applied statistics, and theoretical biology. [Pre04, Rus18, Wan17, Bir07]. Researchers employ a variety of techniques based on differential equations, including stochastic and partial differential equations [Pre04, Wan17]. Some even implement statistical methods such as likelihood estimation and machine learning algorithms to supplement their models [Mic19, Wij20]. Oftentimes, inspiration for modeling animal movement is gained through direct observation, whether through monitoring ant colonies in a nest or radio-collaring elk in a protected forest [Pre04, Rus18].

While we did not have the resources necessary to perform costly observations or the tools to incorporate data into our differential equations, our model is still novel due its focus on attractors in the animal's path. We tailor the model to account for the bee preferences, flower characteristics, distance from attractors, and an objective to eventually return to the hive. Some research focuses on the migration of fish or birds [Bir07], but our model is again unique in its focus on the attractors in a specific path. The value of our work is that it yields results applicable to modeling the movement of animals as well as the movement of humans with real economic implications.

2. Modeling

Primitive Model

As a starting point for modeling the attraction of bees to flowers in a field, we took inspiration from the law of gravity. This law states that the gravitational force exerted on a body is directly proportional to the sum of forces exerted by all other bodies in the universe, where the individual force exerted by any given body is directly proportional to its mass and inversely proportional to the square of the distance between the two objects [Hum23]. We surmised that in our model, the correspondence of a bee's preferences and a flower's unique attributes would play the role of mass in attraction (i.e., the attraction force would be directly proportional to a score of preference correspondence). Each flowers has a vector of attributes to represent features such as size, color, and scent. Each bee has a vector of preferences corresponding to these attributes. By evaluating the norm between these vectors, we find the attraction force of each flower to the bees. Similarly, we can extend this to the case of customers in a store, where each person has a unique set of preferences that may or may not align with the goods they encounter. To extend our model, a retailer could use statistical measures to attempt an accurate estimation of customer preferences.

We note that bees do not zoom exponentially fast towards flowers that catch their attention, thus it was not realistic for the force of attraction to decrease as a function of distance. Observing that bees in nature often achieve a maximum flight velocity that they maintain throughout their travel, we decided to model a particular flower's effect on bee velocity using a logistic function, so that its velocity would be bounded between 0 and some $\mathbf{v}_{max}$. We thus modified the gravitational model to derive the following model of bee position $\mathbf{x_b}$ as a function of the preference correspondence, $\mathbf{p_f}$ and the position $\mathbf{x_f}$ of the attracting flower:

$$G(\mathbf{x_b}, \mathbf{p_f}) = \frac{\alpha \mathbf{p_f}}{\|\mathbf{x_f} - \mathbf{x_b}\|^2} \tag{1}$$

$$\dot{\mathbf{x_b}} = \frac{\mathbf{v}_{max}}{1 + e^{-G(\mathbf{x_b}, \mathbf{p_f})}} \frac{\mathbf{x_f} - \mathbf{x_b}}{\|\mathbf{x_f} - \mathbf{x_b}\|} \tag{2}$$

To test the efficacy of this model, we created a system of one flower placed at the origin and one bee with initial position $\mathbf{x_b} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$. We then generated a numerical solution to the corresponding initial value problem. In this simplistic model, the bee makes a proper beeline to the flower as expected, but in a linear, robotic way that does not represent the trajectory of bees in nature. The exact trajectory taken is shown in the left plot of Figure 1.

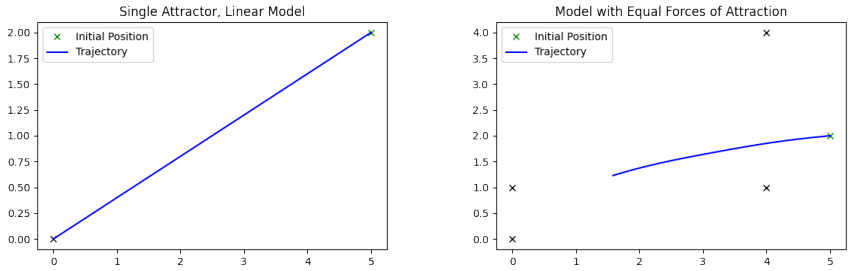


FIGURE 1. Left: The trajectory of a single bee flying from its initial position to the sole attracting flower. Right: A bee starting at its initial position flies and then stops at an unnatural equilibrium–a point without flowers.

In an attempt to introduce curvature in the bee's movement, we added several more flowers and generated another simulation, which is shown in the right plot of Figure 1.

As can be seen, some curvature was introduced. However, after flying in the general direction of the flowers, the bee uncharacteristically settled down and hovered in time, failing to arrive at any flower in the simulation. After some experimentation, we determined that this spot was actually an unforeseen equilibrium because the sum of the forces of attraction was zero

at this position. This unusual result revealed that our assumption that the bee's velocity would be determined by a pure sum of forces of attraction was flawed, introducing unreasonable equilibria into our model.

Intermediate Model

Taking into account the unrealistic linear motion and non-flower equilibrium discovered in the previous model, we updated our model in two important ways. First, to produce movement more characteristic of a bee's flight observed in nature, we introduced a zagging motion to the bee's trajectory. We did so by adding a perturbation to the bee's trajectory in the form of a sine wave perpendicular to the direction of the bee's given velocity as a function of time.

Second, to eliminate the "decision paralysis" effect that we observed in the initial model (the unnatural equilibria where the sum of the forces of attraction of all flowers was zero), we tweaked the model so that the bee always ultimately chooses one flower amidst many attractive flowers. In particular, we set the attraction force of all flowers but the flower with the maximum preference correspondence equal to zero.

These two changes together led to the following updated differential equation for position:

$$\dot{\mathbf{x_b}} = \sum_{\text{f in Field}} ((\mathbb{1}_{\mathbf{p_f}=max\ \mathbf{p}}) \frac{\mathbf{v}_{max}}{1+e^{-G(\mathbf{x_b},\mathbf{p_f})}} \frac{\mathbf{x_f}-\mathbf{x_b}}{\|\mathbf{x_f}-\mathbf{x_b}\|}) + A\dot{\mathbf{x}}_{\mathbf{b}_{prev}} \sin \omega t \tag{3}$$

where A and ω are parameters governing the amplitude and frequency of the zag motion, respectively. The results of this intermediate model produced significantly improved models of bee flight toward an attracting flower, as illustrated in Figure 2.

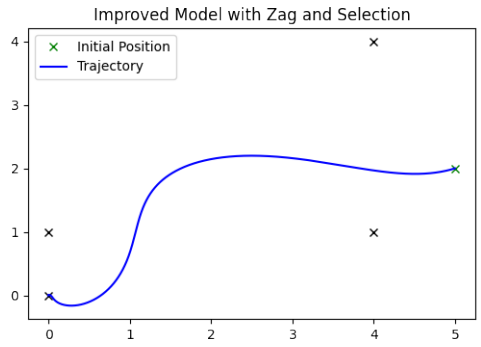


FIGURE 2. A simulation of a bee's trajectory amidst various flowers. Adding natural zag and a selection mechanism to the model resulted in more natural trajectories.

As reflected in the plot of the bee's trajectory under the intermediate model, introducing a sine perturbation to the bee's trajectory resulted in the familiar ambling, casual pattern of a bee in flight. In addition, experimentation gave evidence that this model was free from unexpected equilibria (in other words, there were no points at which a bee stopped at a flowerless location).

Final Model Specifications

While our intermediate model succeeded in producing bee-like movement, it suffered from some limitations as time went on. In particular, once a bee landed at a flower, nothing in our model allowed it to move to any other flowers (i.e., preference was too static). In addition, our model lacked a mechanism for vision; for example, a bee would theoretically pass by a highly attractive flower next to it for a marginally more attractive one significantly farther away.

Our final model more consistently matches environmental and behavioral interpretations. First, we introduce a pervasive attraction to the general goal of returning to a beehive. We also introduce a time limit for agents at individual flowers, after which the flower no longer has any attractive force on that particular bee. This represents how a bee in nature moves to another flower once it has consumed a satifactory amount of nectar. We also adapt our mechanism of flower attraction to be governed by a "vision" factor. This vision factor attracts agents to flowers at a Gaussian rate, allowing flowers to have stronger effects on bees in a given range, and approximately zero effect on flowers "out of sight".

The parameters and equations governing motion in this updated model are listed in Table 1 below. In what follows, n is the total number of attractors, while K is the total number of bees. The model presented herein simulates bee movement, focusing on the interaction between bees and flowers. The key variable, $x_k \in \mathbb{R}^2$, represents the bee's position. The model integrates various environmental parameters as detailed in Table 1 in order to describe the complex dynamics of bee navigation as follows:

$$G_i(x_k, p_k) = \left(e^{-\alpha||f_i - x_k||^2} + b\right)\left(\frac{1}{1 + ||q_i - p_k||^2}\right)\left(\frac{f_i - x_k}{||f_i - x_k||}\right) \tag{4}$$

$$B(x_k, t) = \mathbb{1}(G_{1,\dots,n})\left(\frac{t}{1 + ||h - x_k||}\right)\left(\frac{h - x_k}{||h - x_k||}\right) \tag{5}$$

$$\gamma_i(x_k, t) = \mathbb{1}_{[f_i, T_i]}(x_k, t) \tag{6}$$

$$\gamma^T = [\gamma_1, \gamma_2, \dots, \gamma_n] \tag{7}$$

$$G = [G_1, G_2, \dots, G_n]^T \tag{8}$$

$$\dot{x_k} = V\frac{\gamma^T G + B}{\xi + ||\gamma^T G + B||} \tag{9}$$

Parameter	Description
$h \in \mathbb{R}^2$	Hive position
$f_i \in \mathbb{R}^2$	Attractor position (flower) $i \in \{1, \ldots, n\}$
$p_k \in \mathbb{R}^L$	Preference vector of agent k $\in \{1, \ldots, K\}$
$q_i \in \mathbb{R}^L$	Preference vector of attractor i
α	Vision distance
b	Minimum baseline attraction
T_i	Nectar value (time at attractor)
V	Max speed
ξ	Flight acceleration factor

TABLE 1. Parameters and their Descriptions

The attraction of bee k to flower i is modeled by $G_i(x_k, p_k)$, as defined in Equation 4. This function of the bee's position and preferences comprises a Gaussian-like vision factor, a preference-matching component, and a unit vector in the direction of the flower. We also factor in a persistent pull towards the hive, as given by $B(x_k, t)$ in Equation 5. For this equation, the indicator is activated when the sum of the G_i is greater than one, the middle expression represents an attraction to the hive that grows over time, and finally a unit direction of attraction. It can be shown that

$$||\gamma^T G|| \leq n, \tag{10}$$

which provides an upper bound on the norm of the attractors. This ensures the bees will eventually be more attracted to the hive than the other attractors combined.

The memory indicator $\gamma_i(x_k, t)$, given by Equation 6, determines whether a flower has been visited. This indicator function is active by default, becoming inactive only when the "nectar value" T_i is negative. When an agent is near attractor i, T_i decreases linearly. Equations 7 and 8 define the vectorized notation of the collective memory and attraction across all flowers, denoted as γ^T and G respectively. The overall first-order system for the bee's velocity is governed by the equation $\dot{x_k} = V\frac{\gamma^T G+B}{\xi+||\gamma^T G+B||}$, where V denotes the max speed which bees can fly and ξ is an acceleration factor that allows for non-uniform speed.

This comprehensive framework integrates behavioral and environmental factors to provide a nuanced understanding of bee movement. The equations consider not only the physical dynamics of flight, but also incorporate behavioral elements such as memory and preference, offering a detailed insight into the navigation patterns of bees.

3. Results

With the model specifications above, we find that with sufficient parameter tuning, we obtain a highly effective dynamic model of bee movement over time. Given the parameters

Grid Size	α	b	T_i	V	ξ
100	0.005	0.01	3	500	0.02

along with the random placement of flowers through the field, initial starting positions of bees along the bottom of the grid, and the beehive situated directly at the top, we obtain the plot shown in Figure 3.

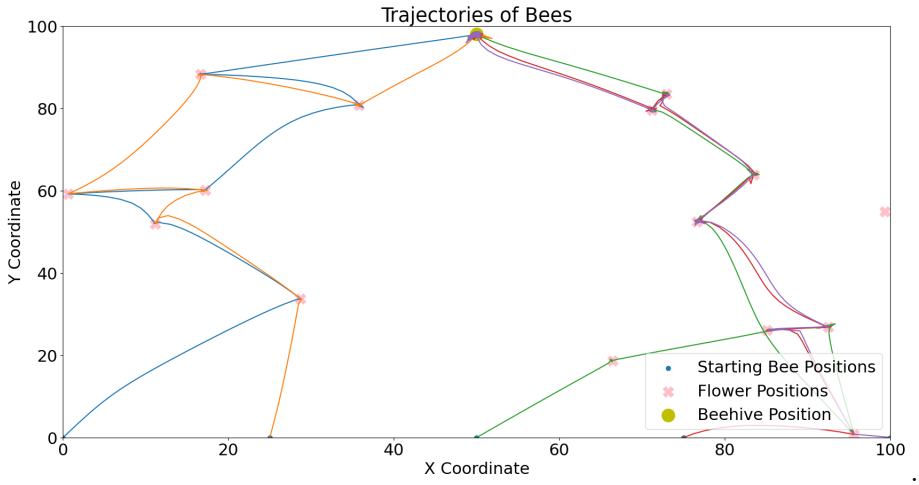


Figure 3. Final model: Several bees visit several attractive flowers in sight on their way to their destination: the hive.

As evidenced by the figure, bees starting at the bottom of the field move from flower to flower, seemingly distracted as they eventually make their way to their original goal of the beehive. Observing the simultaneous evolution of the bees' trajectories demonstrates the primary intent of our model.

Further exploration of the model can be done by exploring vector fields of the attraction force on a particular bee. Given that different preference vectors of bees and flowers alike will result in different vector fields, we randomly construct a single field with varying flower preferences. Our vector field proves particularly useful in uncovering the behavior of our bees after a specific flower has already been visited and highlights the discontinuous change in the bee's trajectory. These vector fields are shown in Figure 4.

This vector field also highlights how the attractors prevent our bee from progressing to its original goal. These distractions may form trapping regions, as seen by the whitespace in the vector fields. Once the attractors are visited, these trapping regions disappear, allowing the agent to progress towards its original goal once more.

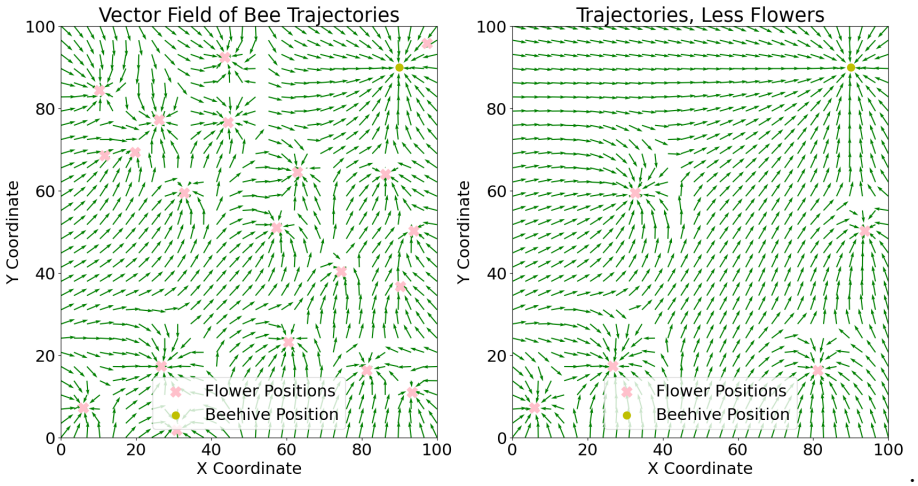


FIGURE 4. Plots of the vector field for the sum attraction force acting on a bee at any given position in the field shown before the bee begins its trajectory (left) and after the bee has visited most of the flowers (right).

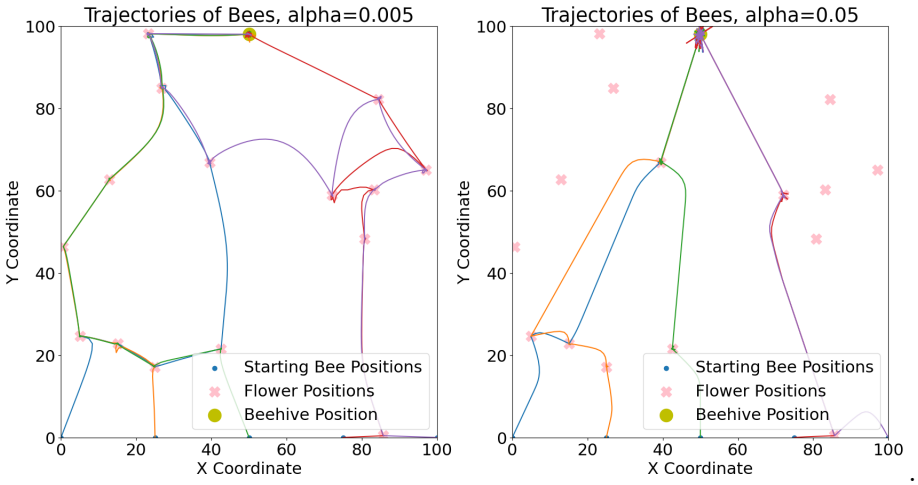


FIGURE 5. Realization of bee trajectories through a field for differing values of α. When α is small, the bees have a wide range of vision and are easily distracted. When α is large, the bees experience narrow vision and consequently become laser-focused on the hive.

Changing parameters affects the path the bees follow. In particular, we can tune the trajectory of the bees to visit more or less flowers by changing

the parameters governing the Gaussian distance function. Figure 5 demonstrates how, by increasing the parameter α by 10-fold, we can narrow the radius of attraction, forcing the bees to ignore flowers further away.

4. Analysis/Conclusions

The techniques employed in modeling the flight patterns of bumblebees navigating a field of attractors were well-suited to capture the essence of the chosen phenomenon. Our group's approach effectively simulated the nuanced dynamics of agent motion amidst multiple enticing destinations. However, with more time and resources, several avenues for refining the model could have been explored, potentially enhancing its accuracy and applicability.

Appropriateness of Methods

The selected methods proved appropriate for capturing the fundamental aspects of bee flight behavior. Leveraging differential equations to model the attraction between bees and flowers effectively represented the intricate interplay between agents and multiple attractors. The introduction of preference vectors and environmental factors provided a comprehensive understanding of bee navigation.

Model Enhancements

Given additional time, several improvements could have been implemented to further enhance the model's fidelity. A periodic beehive factor would allow an agent to start and end at the same destination, like a shopper entering and exiting a store. This factor, initially time-dependent, could evolve to account for daily activities or number of items visited, rendering the model more lifelike.

Insights from the Model

In its current form the model offers valuable insights into various real-world scenarios. As mentioned before, its adaptability extends beyond the study of bees to scenarios such as customer paths in stores, visitor trajectories in theme parks, or tourists navigating cities. Understanding customer preferences and the influential role of proximity, as modeled using the vision factor above, empowers retailers to optimize store layouts. By strategically incorporating secondary attractors while directing attention towards primary objectives, retailers can potentially enhance profits through well-crafted, distraction-driven layouts.

Learning and Future Directions

This project opened a window into the world of mathematical modeling, revealing its collaborative and creative essence. Working as a team to formulate meaningful equations without the structure of a standard homework assignment was eye opening. It allowed us to step into the shoes of mathematicians, crafting our unique path rather than echoing established concepts.

Future iterations of this model could explore the impact of differential preferences, diverse environmental factors, and complex interactions among agents, presenting opportunities for in-depth investigations into behavior-based modeling across diverse domains.

In conclusion, while the model effectively simulated the essential aspects of bee flight behavior, further refinements and extensions could amplify its realism and applicability across various real-world scenarios.

References

[Bir07] B. Birnir. An ode model of the motion of pelagic fish. *Journal of Statistical Physics*, 2007.

[Hum23] Jarvis T. Whitehead J. Humpherys, J. *Volume 4: Modeling with Dynamics and Control*. SIAM, 2023.

[Mic19] Gloaguen P. Blackwell P. Etienne M. Michelot, T. The langevin diffusion as a continuous-time model of animal movement and habitat selection. *Methods in Ecology and Evolution*, (10):1894–1907, 2019.

[Pre04] Ager A. Johnson B. Kie J. Preisler, H. Modeling animal movements using stochastic differential equations. *Environmetrics*, (15):643–357, 2004.

[Rus18] Hanks E. Haran M. Hughes D. Russell, J. A spatially varying stochastic differential equation model for animal movement. *The Annals of Applied Statistics*, 12(2):1312–1331, 2018.

[Wan17] Potts J. Wang, Y. Partial differential equation techniques for analysing animal movement: A comparison of different methods. *Journal of Theoretical Biology*, (416):52–67, 2017.

[Wij20] Eisenhauer E. Shaby B. Hanks E. Wijeyakulasuriya, D. Machine learning for modeling animal movement. *PLoS ONE*, 2020.

all-the-buzz

December 7, 2023

[268]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import solve_ivp
import matplotlib.animation as animation
from matplotlib.animation import FuncAnimation
```

0.1 Generate random positions and attractions for the flowers and bees

[270]:
```
def random_flower_attraction(features=2, num_flowers = 10, max_attraction = 10):
    # Random initial condition, only positive, numbers between 0 and␣
 ↪max_attraction
    return np.random.rand(features, num_flowers) * max_attraction
def random_flower_position(num_flowers = 10, max_position = 10):
    # Random initial condition, only positive, numbers between 0 and␣
 ↪max_position
    return np.random.rand(2, num_flowers) * max_position
def random_bee_position(num_bees = 10, max_position = 10):
    # Random initial condition x and y, only positive coordinates
    return np.random.rand(2, num_bees) * max_position
def random_bee_attraction(features=2, num_bees = 10, max_attraction=10):
    # Random initial condition, only positive, numbers between 0 and␣
 ↪max_attraction
    return np.random.rand(features, num_bees) * max_attraction
```

0.2 Define the distance functions

[275]:
```
def gauss_dist(norms, alpha, c, b):
    # Gaussian distance function
    return c*np.exp(-alpha*(norms**2)) + b


def attraction_to_flower_gaussian(attraction_differences, distance,␣
 ↪distance_norms, alpha=1, c=1, b=0):
    """
    Include gamma array to modulate the attraction based on visited flowers.
    Params:
        mutual_attraction: features x num_flowers
```

```
        mutual_distances: 2 x num_flowers
        gamma: array representing the attraction modulation
        max_velocity: float
    """
    # find the norm of the distance and the attraction
    attraction_norm = np.linalg.norm(attraction_differences, axis=0)
    # find the unit vector of the distance
    velocity_unit_vector = distance / (distance_norms + 1e-8)
    # return the net attraction
    return (velocity_unit_vector * gauss_dist(distance_norms, alpha, c, b)) /␣
↪(1 + attraction_norm)
```

0.3 Define the ODE function

[339]:
```
def ode(t, x_flat, T_i):

    # Reshape x back to its original 2D shape
    x = x_flat.reshape((2, -1))

    # get the distance from the flowers and the beehive
    distance_from_flowers = flower_positions[:, np.newaxis, :] - x[:, :, np.
↪newaxis]
    distance_from_beehive = beehive_position[:, np.newaxis, :] - x[:, :, np.
↪newaxis]

    # get the norm of the distance from the flowers and the beehive
    flower_norms = np.linalg.norm(distance_from_flowers, axis=0)

    # minus for the bees that are at the flowers
    T_i -= (flower_norms < eps) * ((t_final / t_steps) * (1 /␣
↪seconds_at_flower))

    # gamma is 1 if T_i is greater than 0, 0 otherwise
    gamma = T_i > 0

    # get the attraction to the flowers
    attractors = attraction_to_flower_gaussian(attraction_differences,␣
↪distance_from_flowers, flower_norms, alpha, c, b)

    # scale the attraction by gamma to turn off the attraction if the bee has␣
↪visited the flower
    attractors = attractors * gamma

    # sum the attraction vectors
    combined_vector = attractors.sum(axis=2)

    # unit vector in direction of beehive
```

```
    beehive_direction = (distance_from_beehive / (np.linalg.
↪norm(distance_from_beehive, axis=0))).squeeze()

    # scaling factor for beehive direction, function of time and distance from␣
↪beehive
    beehive_scale = ((1 / (np.linalg.norm(distance_from_beehive, axis=0) + 1))␣
↪* (50+t)).squeeze()

    # attraction to the beehive
    beehive_attraction = (beehive_direction * beehive_scale).squeeze()

    # if the norm of the combined vector is greater than 1, set the beehive␣
↪attraction to 0
    if np.linalg.norm(combined_vector, axis=0).any() > 1:
        beehive_attraction = np.zeros_like(beehive_attraction)

    # calculate the velocity of the bees
    velocity = (nu * (combined_vector + beehive_attraction)) / (xsi + np.linalg.
↪norm(combined_vector + beehive_attraction, axis=0))

    # return the velocity for each bee
    return velocity.flatten()
```

0.4 Define constants used for the model

[349]:
```
# number of flowers and bees
flower_qnty = 20
bee_qnty = 5
# max attraction value and max position value
max_attraction = 4
max_position = 100
# number of features for the flowers and bees to be attracted to
features = 5
# time steps and final time
t_steps = 2
t_final = 0.3
t_vals = np.linspace(0,t_final,t_steps)

# model parameters
nu = .2 * max_position # speed of the bees
xsi = 0.02 # nonlinearity of the bees
alpha = 0.5 / max_position # vision radius of the bees
c = 10 # attraction strength of the bees
b = 0.1 # constant attraction of the bees
eps = 0.6  # Proximity threshold for flower visitation
seconds_at_flower =  3 # seconds at flower before leaving
```

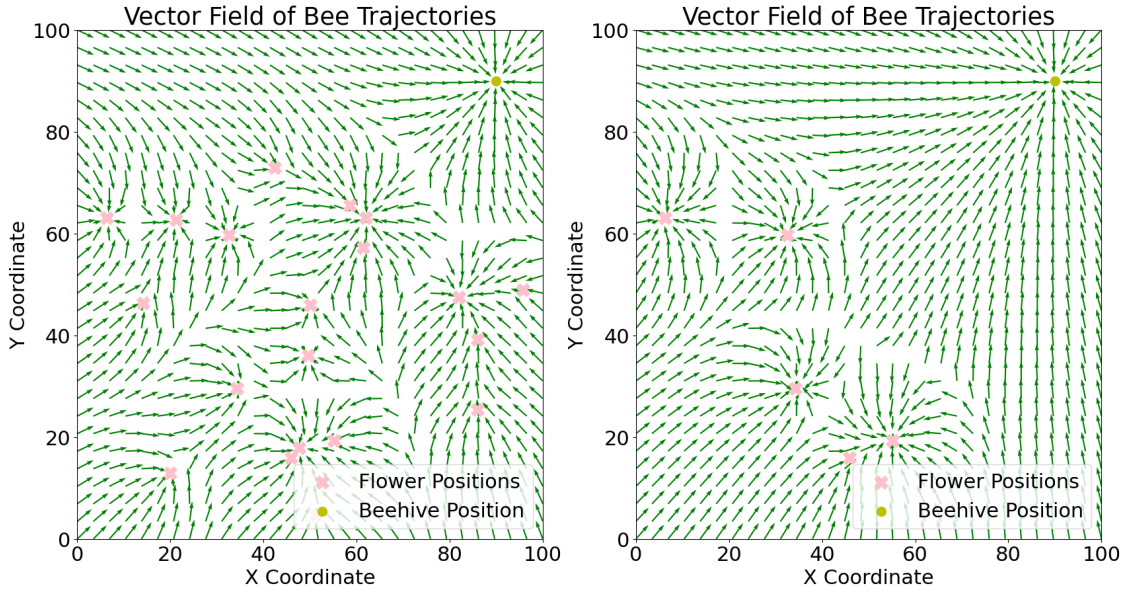
0.5 Calculate and display the initial vector field of the bees

[350]:
```
# grid size for uniform bee distribution
grid_size = 30

# create a grid of positions for the bees
x_positions = np.linspace(0, max_position, grid_size)
y_positions = np.linspace(0, max_position, grid_size)

# create a meshgrid of the positions
xx, yy = np.meshgrid(x_positions, y_positions)
bee_positions = np.vstack((xx.flatten(), yy.flatten()))

# no attraction difference for the bees
bee_preferences = np.zeros((features, bee_positions.shape[1]))

# random initial conditions for the flowers
flower_positions = random_flower_position(num_flowers=flower_qnty,␣
 ↪max_position=max_position)
flower_attraction = random_flower_attraction(num_flowers=flower_qnty,␣
 ↪features=features, max_attraction=max_attraction)

# random initial conditions for the bees
near_corner = max_position - (0.1 * max_position)
beehive_position = np.array([near_corner, near_corner]).reshape(-1, 1)
bee_qnty = bee_positions.shape[1]

attraction_differences = bee_preferences[:, :, np.newaxis] - flower_attraction[:
 ↪, np.newaxis, :]
T_i = np.ones((bee_qnty, flower_qnty))
```

[351]:
```
solution = solve_ivp(ode, (0,t_final), bee_positions.flatten(), t_eval=t_vals,␣
 ↪args=(T_i,), dense_output=True)

# Extract the solution
trajectories = solution.y

num_bees = bee_qnty
num_time_points = trajectories.shape[1]
trajectories_reshaped = trajectories.reshape((2, num_bees, num_time_points))

# 2 subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))


arrow_color = 'green'
```

```
def plot_ax(ax, arrow_color, title, trajectories_reshaped, bee_positions,␣
 ↪flower_positions, beehive_position, max_position, num_bees, num_time_points):
    for i in range(num_bees):
        start_x = bee_positions[0, i]
        start_y = bee_positions[1, i]
        end_x = trajectories_reshaped[0, i, -1]
        end_y = trajectories_reshaped[1, i, -1]

        ax.quiver(start_x, start_y, end_x - start_x, end_y - start_y,
                  color=arrow_color, angles='xy', scale_units='xy', scale=1.
 ↪5, width=0.003)

    ax.scatter(flower_positions[0, :], flower_positions[1, :], marker='X',␣
 ↪color='pink', s=200, label='Flower Positions')
    ax.scatter(beehive_position[0, :], beehive_position[1, :], marker='o',␣
 ↪color='y', s=100, label='Beehive Position')
    ax.set_xlabel('X Coordinate')
    ax.set_ylabel('Y Coordinate')
    ax.set_title('Vector Field of Bee Trajectories')
    ax.set_xlim(0, max_position)
    ax.set_ylim(0, max_position)
    ax.legend(loc='lower right')

plot_ax(ax1, arrow_color, 'Vector Field of Bee Trajectories',␣
 ↪trajectories_reshaped, bee_positions, flower_positions, beehive_position,␣
 ↪max_position, num_bees, num_time_points)

# remove a flower from the flower positions
flower_positions = flower_positions[:, :-15]
flower_attraction = flower_attraction[:, :-15]
flower_qnty -= 15

attraction_differences = bee_preferences[:, :, np.newaxis] - flower_attraction[:
 ↪, np.newaxis, :]  # Shape -> (num_features, num_bees, num_flowers)
T_i = np.ones((bee_qnty, flower_qnty))

solution = solve_ivp(ode, (0,t_final), bee_positions.flatten(), t_eval=t_vals,␣
 ↪args=(T_i,), dense_output=True)

# Extract the solution
trajectories = solution.y

# Reshape the trajectories to (2, num_bees, num_time_points)
num_bees = bee_qnty
num_time_points = trajectories.shape[1]
trajectories_reshaped = trajectories.reshape((2, num_bees, num_time_points))
```

```
plot_ax(ax2, arrow_color, 'Vector Field of Bee Trajectories',␣
 ↪trajectories_reshaped, bee_positions, flower_positions, beehive_position,␣
 ↪max_position, num_bees, num_time_points)

plt.show()
```



0.6 Define constants used for the model

[371]:
```
# number of flowers and bees
flower_qnty = 15
bee_qnty = 5
# max attraction value and max position value
max_attraction = 4
max_position = 100
# number of features for the flowers and bees to be attracted to
features = 5
# time steps and final time
t_steps = 400
t_final = t_steps / 30
t_vals = np.linspace(0,t_final,t_steps)

# model parameters
nu = .5 * max_position # speed of the bees
xsi = 0.02 # nonlinearity of the bees
alpha = 0.5 / max_position # vision radius of the bees
c = 10 # attraction strength of the bees
```

```
b = 0.1 # constant attraction of the bees
eps = 0.6  # Proximity threshold for flower visitation
seconds_at_flower =  3 # seconds at flower before leaving
```

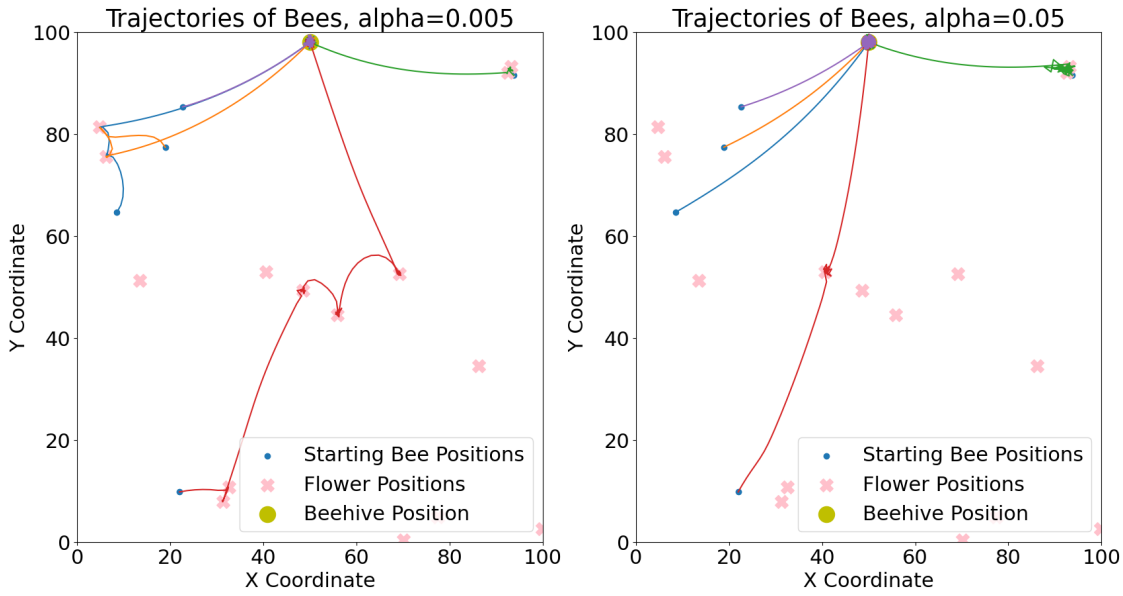
0.7 Initial conditions of the bees and flowers

[358]:
```
# random positions for the flowers and bees
flower_positions = random_flower_position(num_flowers=flower_qnty,␣
 ↪max_position=max_position)
bee_positions = random_bee_position(num_bees=bee_qnty,␣
 ↪max_position=max_position)

# # have bees start at the bottom edge of the plot, evenly spaced
# bee_positions = np.array([np.linspace(0, max_position, bee_qnty), np.
 ↪zeros(bee_qnty)]).reshape(2, -1)

# random preferences for the bees and attraction for the flowers
bee_preferences = random_bee_attraction(features=features, num_bees=bee_qnty,␣
 ↪max_attraction=max_attraction)
flower_attraction = random_flower_attraction(num_flowers=flower_qnty,␣
 ↪features=features, max_attraction=max_attraction)

# location of the beehive
beehive_position = np.array([max_position/2, max_position - (0.02 *␣
 ↪max_position)]).reshape(-1, 1)
```

0.8 Calculate and display the trajectories of the bees

[372]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

def plot_ax(ax, trajectories, bee_positions, flower_positions,␣
 ↪beehive_position, max_position, alpha):

    num_time_points = trajectories.shape[1]
    trajectories_reshaped = trajectories.reshape((2, num_bees, num_time_points))

    for i in range(bee_qnty):
        # Extracting the x and y coordinates of the i-th bee
        x = trajectories_reshaped[0, i, :]
        y = trajectories_reshaped[1, i, :]
        ax.plot(x, y)

    ax.scatter(bee_positions[0, :], bee_positions[1, :], label='Starting Bee␣
 ↪Positions')
    # plot the flowers
```

```
    ax.scatter(flower_positions[0, :], flower_positions[1, :], marker='X',␣
 ↪color='pink', s=200, label='Flower Positions')

    # plot the beehive as large yellow dot
    ax.scatter(beehive_position[0, :], beehive_position[1, :], marker='o',␣
 ↪color='y', s=300,label='Beehive Position')

    ax.set_xlabel('X Coordinate')
    ax.set_ylabel('Y Coordinate')
    ax.set_title(f'Trajectories of Bees, alpha={alpha}')
    ax.set_xlim(0,max_position)
    ax.set_ylim(0,max_position)
    # put the legend in the top left corner
    ax.legend(loc='lower right')

attraction_differences = bee_preferences[:, :, np.newaxis] - flower_attraction[:
 ↪, np.newaxis, :]  # Shape -> (num_features, num_bees, num_flowers)
T_i = np.ones((bee_qnty, flower_qnty))

solution = solve_ivp(ode, (0,t_final), bee_positions.flatten(), t_eval=t_vals,␣
 ↪args=(T_i,), dense_output=True)

# Extract the solution
trajectories = solution.y
num_bees = bee_qnty

plot_ax(ax1, trajectories, bee_positions, flower_positions, beehive_position,␣
 ↪max_position, alpha)

alpha = 5 / max_position  # Parameter for the ODE

attraction_differences = bee_preferences[:, :, np.newaxis] - flower_attraction[:
 ↪, np.newaxis, :]  # Shape -> (num_features, num_bees, num_flowers)
T_i = np.ones((bee_qnty, flower_qnty))

solution = solve_ivp(ode, (0,t_final), bee_positions.flatten(), t_eval=t_vals,␣
 ↪args=(T_i,), dense_output=True)

# Extract the solution
trajectories = solution.y

plot_ax(ax2, trajectories, bee_positions, flower_positions, beehive_position,␣
 ↪max_position, alpha)
```



0.9 Animate the trajectory

[373]:
```
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlim(-5, max_position + 5)
ax.set_ylim(-5, max_position + 5)
ax.set_xlabel('X Coordinate')
ax.set_ylabel('Y Coordinate')
ax.set_title('Trajectories of Bees (Gaussian)')

bee_lines = [ax.plot([], [])[0] for i in range(num_bees)]

ax.scatter(bee_positions[0, :], bee_positions[1, :], label='Starting Bee␣
 ↪Positions')
ax.scatter(flower_positions[0, :], flower_positions[1, :], marker='x',␣
 ↪color='r', label='Flower Positions')
ax.scatter(beehive_position[0, :], beehive_position[1, :], marker='o',␣
 ↪color='y', s=100,label='Beehive Position')
ax.legend()

def animate(frame):
    # Update the lines for each bee
    for i, line in enumerate(bee_lines):
        line.set_data(trajectories_reshaped[0, i, :frame],␣
 ↪trajectories_reshaped[1, i, :frame])
ani = FuncAnimation(fig, animate, frames=num_time_points, interval=20)
plt.close(fig)
```

[375]:
```
ani.save('final_animation.mp4', writer='ffmpeg')
```