1 Model and simulate the flocking behavior of birds

1.1 What

- In your report, write down the system of ODEs including all forcing terms. Be sure to define each forcing term.
- Implement this system of ODEs using RK4 and experiment with different parameters and different numbers of birds.
- Discuss how the flock diameter changes over time and for different scenarios.

1.2 RK4

First, we implemented **RK4 function** to return updated vector **y** from time **t** using **dt** as follows:

```
def RK4(f, y, t, dt, food_flag, alpha, gamma_1, gamma_2, kappa,
    rho, delta):
    k1 = f(y, t, food_flag, alpha, gamma_1, gamma_2, rho, delta)
    k2 = f(y+(dt*0.5)*k1, t+(dt*0.5), food_flag, alpha, gamma_1,
        gamma_2, kappa, rho, delta)
    k3 = f(y+(dt*0.5)*k2, t+(dt*0.5), food_flag, alpha, gamma_1,
        gamma_2, kappa, rho, delta)
    k4 = f(y+dt*k3, t+dt, food_flag, alpha, gamma_1, gamma_2,
        kappa, rho, delta)
    y = y + (dt/6) * (k1 + 2*k2 + 2*k3 + k4)
    return y
```

1.3 Forcing terms

Assume that we have a flock of N birds.

$B(t)$ will correspond to a length **N** vector with 2 columns, where the **k-th** row of **B** will be the coordinates of bird **k**. We denote this **k-th** row as $B_k(t)$, which equals the (x,y) location of the **k-th** bird at time **t**.

$f(t, B(t))$ represents all the forcing terms, i.e., the right-hand-side of the ODE, and this term governs the movement of your birds. There are many things may influence the birds, so we have come up to calculate $f(t, B(t))$ as the formula:

$$f(t, B(t)) = F_1^{food}(t, B(t)) + F_k^{follow}(t, B(t)) + F_k^{fl}(t, B(t)) + F_k^{fl}(t, B(t)) + F_k^{repl}(t, B(t)) \tag{1}$$

Each forcing term will be discussed in following subsections.

1.3.1 $F_1^{food}(t, B(t))$

This is the force that represents the desire of birds to follow food. Assume $B_1(t)$ be the location of the bird leader who is interested in eating food at the (x_C, y_C) coordinate.
As $\gamma_1 > 0$, the bird leader $B_1(t)$ is encouraged to fly towards the food and reduce its distance to $C(t)$. If $x_C > x_1$, it means that the bird leader will fly towards the right hand side of ODE system.
Therefore, $F_1^{food}(t, B(t))$ is calculated by the difference of the bird leader's location to the food source:

$$F_1^{food}(t, B(t)) = \gamma_1(C(t) - B_1(t)) \tag{2}$$

- $food\_flag$ is the flag set for food: 0 or 1, depending on the food location.
- γ_1 = Food coefficient to scale the following distance of the lead bird
- $C(t)$ is the function that returns the coordinates (x,y) of the food source.
- α = Size of food path in C(t)

Here is the code for $F_1^{food}(t, B(t))$:

```
def RHS(y, t, food_flag, alpha, gamma_1, gamma_2, kappa, rho,
    delta):
    N = y.shape[0]
    f = zeros_like(y)

    if (food_flag == 0):
        C = (0.0, 0.0)
    elif (food_flag == 1):
```

```
        C = (sin(alpha*t), cos(alpha*t))

    f_food = gamma_1*(C - y[0,:])
```

1.3.2 $F_k^{follow}(t, B(t))$

This is the force that represents the desire of birds to follow the bird leader. For this term, we find the difference of the lead birds location to the other birds in the flock.

$$F_k^{follow}(t, B(t)) = \gamma_2(B_1(t) - B_k(t)), (k = 1, 2, 3..., N) \tag{3}$$

- γ_2 = Follow coefficient to indicate how close the following should be

So, in **RHS function**, we defined $F_k^{follow}(t, B(t))$ as:

```
    f_follow = gamma_2*(y[0,:] - y)
```

1.3.3 $F_k^{fl}(t, B(t))$

This flocking force ensures the safety of the flock if the predator attacks by keeping the birds as close as possible to the middle of the flock. So, we first need to define $\overline{B}(t)$ which is the (x,y) coordinate of the mass center.

$$\overline{B}(t) = \sum_{k=1}^{N} = \frac{B_k(t)}{N} \tag{4}$$

Then, for $F_k^{fl}(t, B(t))$, we find the difference between mass center coordinate and each bird's coordinate except the bird leader. This will make sure that the birds are within some boundary of flocking.

$$F_k^{fl}(t, B(t)) = \kappa(\overline{B}(t) - B_k(t)), (k = 2, 3..., N) \tag{5}$$

- κ = Flocking force is the force that keeps the birds together

```
    B_bar = sum(y, axis=0)/N
    f_fl = kappa*(B_bar - y)
    f_fl[0] = 0
```

1.3.4 $F_{k,j}^{rep}(t, B(t))$

Consider the case the birds may slam into each other when they are being too close, we will need a strong repelling force $F_{k,j}^{rep}(t, B(t))$. We assume that the k-th bird B_k is repelled by its 5 closest neighbors, and l_j^k denotes the j-th closest bird to bird k.

$$F_k^{repl}(t, B(t)) = \sum_{i=1}^{5} \rho \frac{(B_{k,j}(t) - (B_{l_i^k,j}(t))}{((B_{k,j}(t) - B_{l_i^k,j}(t))^2 + \delta}, (k = 2, 3..., N) \tag{6}$$

- ρ = Repelling Force magnitude
- δ = Repelling scale

To implement the formula 6, we first initialized an array called **arr_dist** which stores all distance values between other birds and bird k. Sorting this array, we get bird k as the first element and the next 5 closest birds in **arr_d**. Then, we looped over each value in the new sorted array to compute $F_{k,j}^{rep}(t, B(t))$.

```
    arr_dist = zeros((N,2))
    f_rep = zeros_like(y)
    for k in range(1,N):
        for i in range (0,N):
            arr_dist[i] = (i, sqrt((y[i,1] - y[k,1])**2 + (y[i,0] -
                y[k,0])**2))
        arr_d = argsort(arr_dist, axis = 0)

        for j in arr_d[1:6]:
            f_rep[k,:] += rho*((y[k,:] - y[j[1],:]) / ((y[k,:] -
                y[j[1],:])**2 + delta))
```

1.4 Implement this system of ODEs using RK4 different numbers of birds.

```
# Set up problem domain
t0 = 0.0       # start time
T = 10.0       # end time
nsteps = 75    # number of time steps
```

```
# Task: Experiment with N, number of birds
N = 10         # Try N = 30, 100
# Task: Experiment with the problem parameters, and understand how each parameter affects the system
dt = (T - t0) / (nsteps-1.0)
gamma_1 = 2.0
gamma_2 = 8.0
alpha = 0.4
kappa = 4.0
rho = 2.0
delta = 0.5
food_flag = 0  # food_flag == 0: C(x,y) = (0.0, 0.0)
               # food_flag == 1: C(x,y) = (sin(alpha*t), cos(alpha*t))
# Intialize problem
y = random.rand(N,2) # This is the state vector of each Bird's position. The k-th bird's position is (y[k,0], y[k,1])
flock_diam = zeros((nsteps,))
t = t0
    for step in range(nsteps):
        y = RK4(RHS, y, t, dt, food_flag, alpha, gamma_1, gamma_2, kappa, rho, delta)
        B_bar = sum(y, axis = 0)/N

        distance = 0
        for i in range(N):
            distance += sqrt((y[i,1] - B_bar[1])**2 + (y[i,0] - B_bar[0])**2)
            ave_dist = distance / (N)
        flock_diam[step] = ave_dist*2
        t += dt
```

- food_flag = 0 which mean variable C (food source) coordinate will be constant at x = 0 and y = 0. All the bird will be around the food source.
- The variable alpha will determined difference the food source path with food_flag = 1.
- Gamma_1 determined the closer distance between the leader fly toward

the food as gamma_1 increase.

- Gamma_2 determined the closer other birds follow the leader as gamma_2 increase.
- Kappa determined the closer distance between all the bird (except the leader) and the mass center as kappa increase.
- Rho determined how strong the repelling force between the bird k and 5 closest birds.
- Delta also determined the repelling force between the bird k and 5 closest birds as delta decrease the force will be stronger.

We calculated the sum of all distance between the birds and the center of mass (Bbar). Then find the average distance to plot in each step. The flock diameter is the average distance between all the birds and the mass center. As we increase number of birds, the flock diameter will more steady.

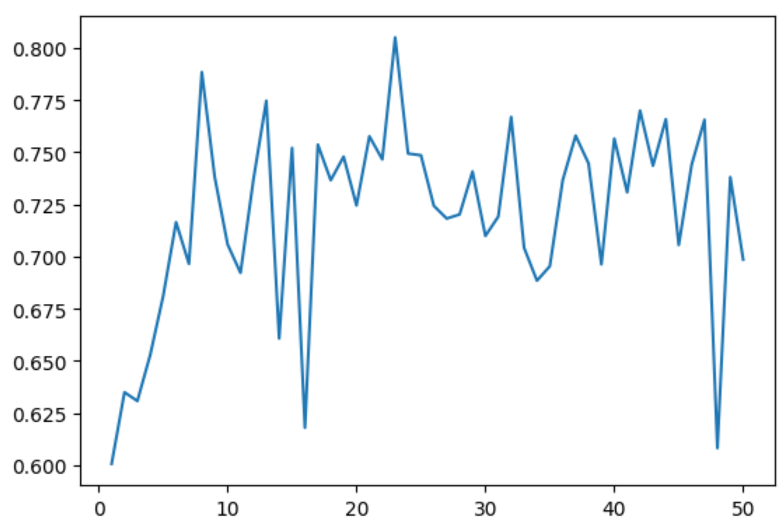

Figure 1: Flock Diameter with N = 10

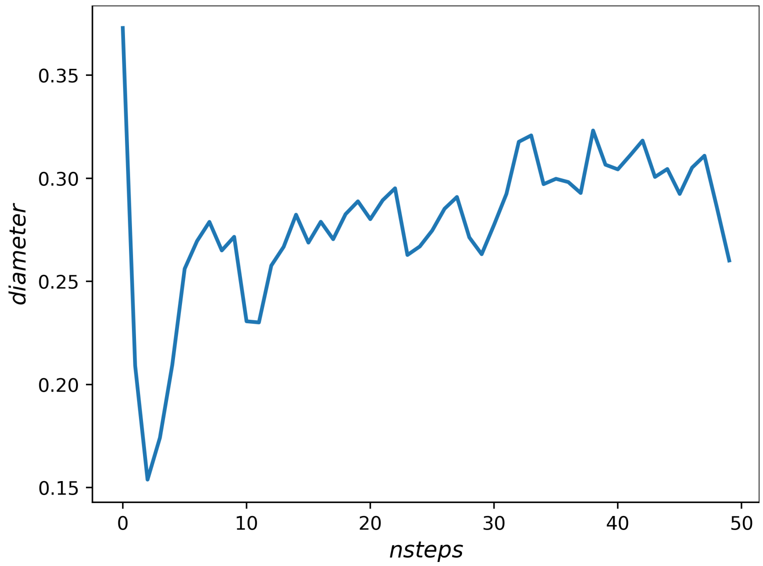

Figure 2: Flock Diameter with N = 30

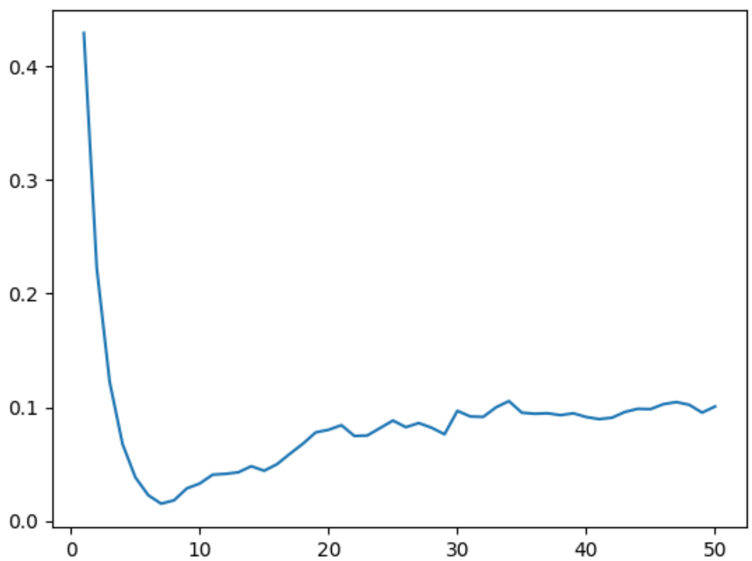

Figure 3: Flock Diameter with N = 100

1.5 Why

We generated ODE system and created movies in time step. Figure 1, 2, 3 showed flock diameter when the number of birds is 10, 30, and 100 respectively. We observed that the flock bird maintains a linear regression line and the data fluctuates due to the size of the bird's position and how much they want to fly with the flock but also have to push back from the leader.

2 The smelly bird and the predator

2.1 What

Implement an ODE for a smelly bird and a predatory bird and adjust to the flock respectively.

2.2 How

We implemented two other ODEs for smelly bird and predatory bird. Based on the flags as true or false, other birds can recognize which one is the smelly bird or predator. In this assignment, we assumed the **29-th** bird is **smelly bird** and the **30-th** bird is the **predator**.

We created another the force call **f_sb** which represents the relationship between smelly bird and others. This force acts similar to repelling force but in this case all the bird will avoid getting close to the smelly bird.

We also created another force for predator called **f_p**. This force is similar to the follow force. In this case, the predator will try to get close to the leader and all other bird will stay away from predator.

The RHS will check the ODE conditions for **f_sb** and **f_p** and compute the function for RK4.

```
    arr_dist = zeros((N,2))
    f_rep = zeros_like(y)
    for k in range(1,N):
        for i in range (0,N):
            arr_dist[i] = (i, sqrt((y[i,1] - y[k,1])**2 + (y[i,0] - y[k,0])**2))
        arr_d = argsort(arr_dist, axis = 0)

        if ((smelly_bird and k == 29) or (predator and k == 30)):
            continue
        for j in arr_d[1:6]:
            f_rep[k,:] += rho*((y[k,:] - y[j[1],:]) / ((y[k,:] - y[j[1],:])**2 + delta))
```

```
    f_sb = zeros_like(y)
    if smelly_bird:
        f_sb = rho*((y - y[29,:]) / ((y - y[29,:])**2 + delta))*3

    f_p = zeros_like(y)
    if predator:
        y[29,0] = -2
        y[29,1] = 2
        predator_loc = (y[29,0], y[29,1])
        f_p = gamma_2*(y - predator_loc)/6
        f_p[30,:] = gamma_2*(y[0,:] - predator_loc)/5

# Begin writing movie frames
with writer.saving(fig, "flock_movie.mp4", dpi=1000):

    # First frame
    pp.set_data(y[1:,0], y[1:,1])
    rr.set_data(y[0,0], y[0,1])
    if smelly_bird:
        sb.set_data(y[29,0], y[29,1])
    if predator:
        pr.set_data(-2, 2)
    writer.grab_frame()

    t = t0
    for step in range(nsteps):

        # Task: Fill in the code for the next two lines
        y = RK4(RHS, y, t, dt, food_flag, alpha, gamma_1, gamma_2,
            kappa, rho, delta)
        B_bar = sum(y, axis = 0)/N

        distance = 0
        if predator:
            for i in range(N-1):
                distance += sqrt((y[i,1] - B_bar[1])**2 + (y[i,0] -
                    B_bar[0])**2)
            ave_dist = distance / (N-1)
        else:
```

```
            for i in range(N):
                distance += sqrt((y[i,1] - B_bar[1])**2 + (y[i,0] - B_bar[0])**2)
            ave_dist = distance / (N)
        flock_diam[step] = ave_dist*2
        t += dt

        # Movie frame
        pp.set_data(y[:,0], y[:,1])
        rr.set_data(y[0,0], y[0,1])
        if smelly_bird:
            sb.set_data(y[29,0], y[29,1])
        if predator:
            pr.set_data(y[30,0], y[30,1])
        writer.grab_frame()
```

Here are flock diameters when there is a smelly bird and a predator bird:

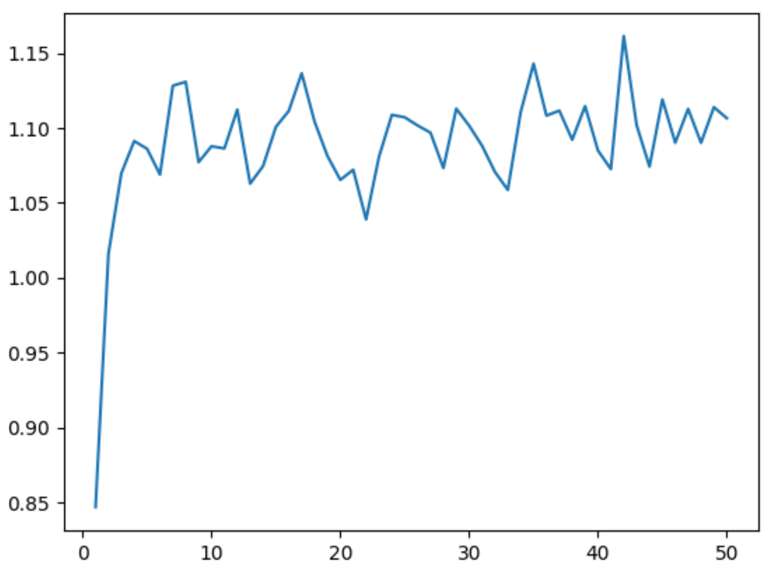

Figure 4: Flock Diameter with smelly bird

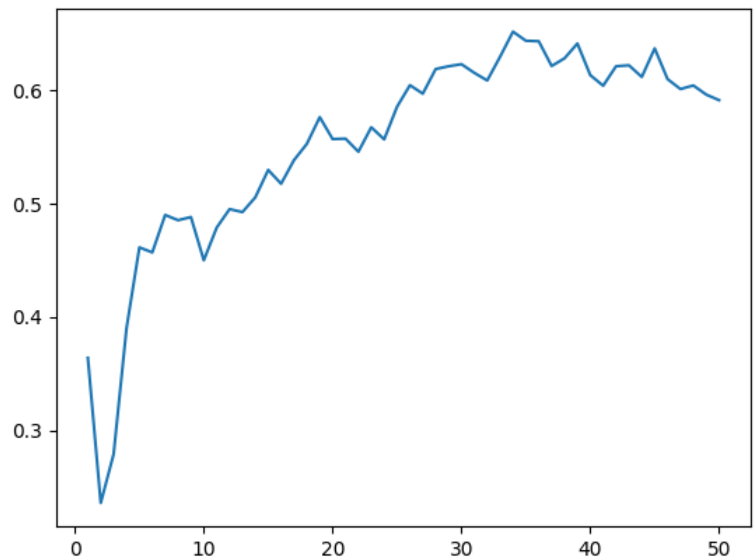

Figure 5: Flock Diameter with predator

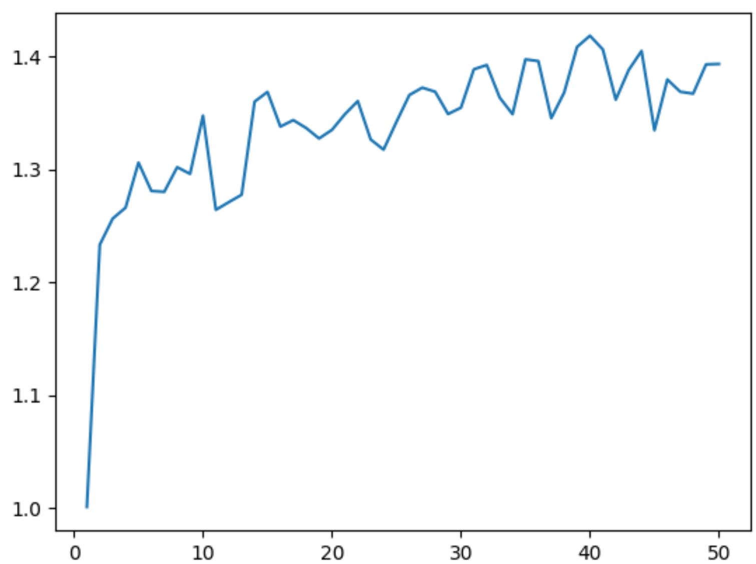

Figure 6: Flock Diameter with smelly bird and predator

2.3 Why

We can see that when a smelly bird appears, the flock of birds will create a distance to keep away from the smelly bird. This distance will be larger than **f_rep** (which is not to let the flock birds collide). And in the case of a predator appears, flock of birds will create distance from the predator bird. When the predator approaches the flock, the flock will move away.