```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython.display import HTML
np.random.seed(1)
vmax = 1.2
L = 60.0
N_patches = 3
patch_pos = [np.random.uniform(-5, 5, 2) for _ in range(3)]
patch_vel = [np.random.randn(2) * 0.2 for _ in range(3)]
P = [60.0, 60.0, 60.0]
split_threshold = 20.0
prey_growth = 4.0
prey_diffusion = 0.10
N_sharks = 25
sharks = np.random.uniform(-L, L, (N_sharks, 2))
vel = np.zeros_like(sharks)
D = 0.3
chi = 1.8
memory = 0.85
capture_radius = 0.5
hunt_rate = 0.005
a = 0.8
alpha0 = 0.05
#beta = 0.02
delta = 0.12
gamma = 0.10
sigma = 0.6
lam = 1.0
E0 = 80.0
eps = 0.3
field_noise = 0.25
occlusion_length = 6.0
h = 0.02
#ke = 0.05
dt = 0.5
steps = 1000
K = 100
noise_P = 0.08
noise_R = 0.06
noise_E = 0.05
R = [float(N_sharks)]
def prey_field(x):
    E = 0.0
    for j in range(len(patch_pos)):
        r = x - patch_pos[j]
        r2 = np.sum(r**2) + eps ** 2
        E += P[j] * E0 / r2
    return E
```

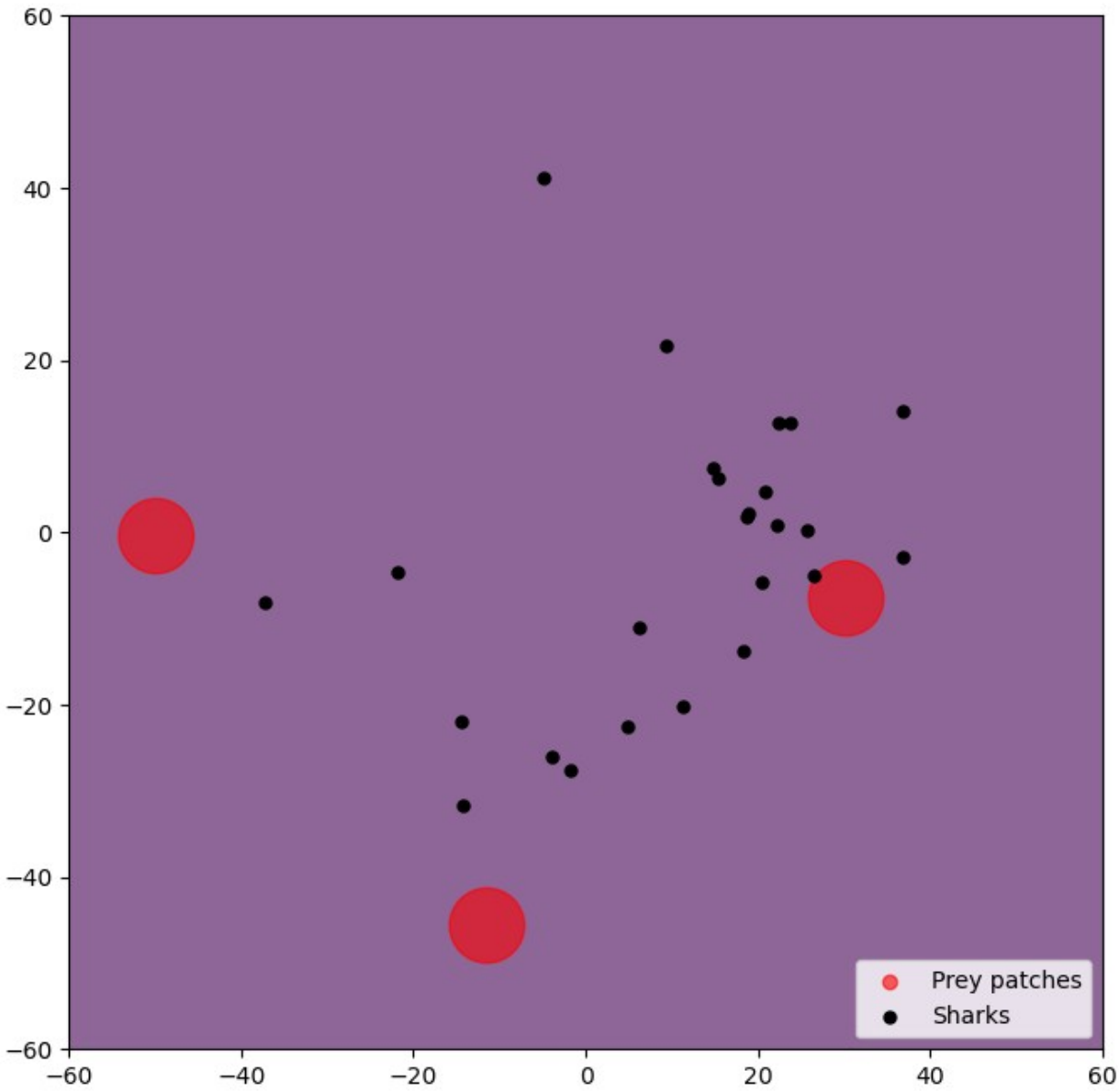
```
def grad_prey_field(x):
    g = np.zeros(2)
    for j in range(len(patch_pos)):
        r = x - patch_pos[j]
        r2 = np.sum(r ** 2) + eps ** 2
        g += -2 * P[j] * E0 * r / (r2 ** 2)
    return g
def predator_threat(x):
    g = np.zeros(2)
    for s in sharks:
        r = x - s
        r2 = np.sum(r ** 2) + eps ** 2
        g += r / r2
    return g
grid_n = 60
xs = np.linspace(-L, L, grid_n)
ys = np.linspace(-L, L, grid_n)
X, Y = np.meshgrid(xs, ys)
#R = np.zeros(steps)
#E = np.zeros(steps)
#P = [40.0]
#R = [10.0]
#E = [0.0]
#time = [0.0]
fig, ax = plt.subplots(figsize = (8, 8))
ax.set_xlim(-L, L)
ax.set_ylim(-L, L)
ax.set_aspect("equal")
field_img = ax.imshow(np.zeros((grid_n, grid_n)), extent = (-L, L, -L,
L), origin="lower", cmap="viridis", alpha = 0.6)
prey_scatter = ax.scatter([], [], c="red", alpha=0.6, label="Prey
patches")
shark_scatter = ax.scatter(sharks[:,0], sharks[:,1], c="black", s=25,
label = "Sharks")
ax.legend(loc="lower right")
#ax.set_xlabel("Time")
#ax.set_ylabel("Population/ Bioelectric Signal")
#prey_line, = ax.plot([], [], lw=2, label = "Prey")
#pred_line, = ax.plot([], [], lw=2, label = "Predator")
#bio_line, = ax.plot([], [], lw=2, linestyle ="--", label="Bioelectric
Signal")
#ax.legend()
####bio_line.set_data([], [])
    #return prey_line, pred_line, bio_line
def update(frame):
    #Pn = P[-1]
    #Rn = R[-1]
    #En = E[-1]#ke * Pn
    #alpha = alpha0 * (1 + (beta * En)/(1 + beta * En))
```

```
    global patch_pos, patch_vel, P
    #patch_pos += patch_vel * dt
    #patch_vel += prey_diffusion * np.random.randn(*patch_vel.shape) *
np.sqrt(dt)
    for i in range(N_sharks):
        g = grad_prey_field(sharks[i])
        noise = np.sqrt(2 * D * dt) * np.random.randn(2)
        vel[i] = memory * vel[i] + chi * g * dt + noise
        speed = np.linalg.norm(vel[i])
        if speed > vmax:
            vel[i] *= vmax / speed
        sharks[i] += vel[i]
        for d in range(2):
            if abs(sharks[i, d]) > L:
                sharks[i, d] = np.sign(sharks[i, d]) * L
                vel[i, d] *= -1
    for j in range(len(patch_pos)):
        flee = predator_threat(patch_pos[j])
        patch_vel[j] += 0.6 * flee * dt
        patch_vel[j] += prey_diffusion * np.random.randn(2) *
np.sqrt(dt)
        patch_pos[j] += patch_vel[j] * dt
        for d in range(2):
            if abs(patch_pos[j][d]) > L:
                patch_pos[j][d] = np.sign(patch_pos[j][d]) * L
                patch_vel[j][d] *= -1
    for j in range(len(patch_pos)):
        dist = np.linalg.norm(sharks - patch_pos[j], axis = 1)
        attackers = np.sum(dist < capture_radius)
        P[j] = max(P[j] - hunt_rate * attackers * P[j] * dt, 0.0)
    new_pos, new_vel, new_P = [], [], []
    for j in range(len(P)):
        if P[j] < split_threshold and P[j] > 5:
            offset = np.random.randn(2)
            offset /= np.linalg.norm(offset)
            new_pos += [patch_pos[j] + offset, patch_pos[j] - offset]
            new_vel += [np.random.randn(2) * 0.2, np.random.randn(2) *
0.2]
            new_P += [P[j]/2, P[j]/2]
        else:
            new_pos.append(patch_pos[j])
            new_vel.append(patch_vel[j])
            new_P.append(P[j])
    patch_pos[:] = new_pos
    patch_vel[:] = new_vel
    P[:] = new_P
    for j in range(len(P)):
        P[j] += prey_growth * P[j] * (1 - P[j]/K) * dt
    field = np.zeros_like(X)
```

```
    for j in range(len(patch_pos)):
        r2 = (X - patch_pos[j][0]) ** 2 + (Y - patch_pos[j][1]) ** 2 +
eps ** 2
        field += P[j] * E0 / r2
    field_img.set_data(field)
    prey_scatter.set_offsets(np.array(patch_pos))
    prey_scatter.set_sizes([10 * p + 20 for p in P])
    shark_scatter.set_offsets(sharks)
    return field_img, prey_scatter, shark_scatter
    #dE_det = (sigma * Pn - lam * En) * dt
    #dE_sto = noise_E * En * np.sqrt(dt) * np.random.randn()
    #En_new = max(En + dE_det + dE_sto, 0)
    #alpha = alpha0 * (1 + En_new / (E0 + En_new))
    #predation = alpha * Pn * Rn/ (1 + h * Pn)
    #dP_det = (a * Pn * (1 - (Pn/K)) - predation) * dt
    #dP_sto = noise_P * Pn * np.sqrt(dt) * np.random.randn()
    #P_new = max(Pn + dP_det + dP_sto, 0)
    #dR_det = (delta * predation  - gamma * Rn) * dt
    #dR_sto = noise_R * Rn * np.sqrt(dt) * np.random.randn()
    #R_new = max(Rn + dR_det + dR_sto, 0)
    #P.append(P_new)
    #R.append(R_new)
    #E.append(En_new)
    #time.append(time[-1] + dt)
    #prey_line.set_data(time, P)
    #pred_line.set_data(time, R)
    #bio_line.set_data(time, E)
    #ax.set_xlim(0, time[-1])
    #return prey_line, pred_line, bio_line
ani = FuncAnimation(fig, update, frames=steps, interval=40,
blit=False)
HTML(ani.to_jshtml())
#plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
Animation size has reached 20975497 bytes, exceeding the limit of
20971520.0. If you're sure you want a larger animation embedded, set
the animation.embed_limit rc parameter to a larger value (in MB). This
and further frames will be dropped.
```

```
<IPython.core.display.HTML object>
```


60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60
Prey patches
Sharks