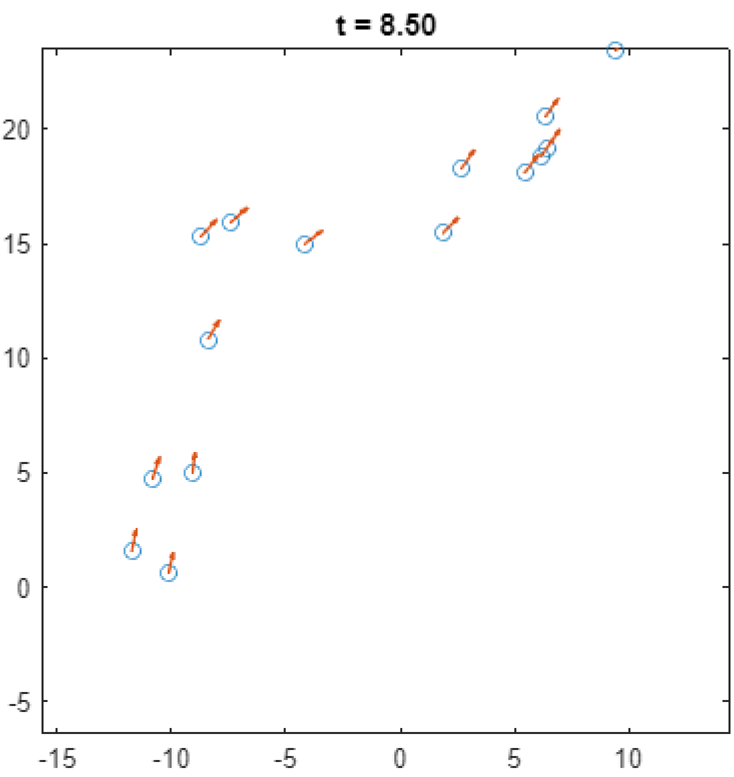
```
% PLOTTING
win_scale = 15;
window = [-win_scale, win_scale, -win_scale, win_scale];
movieFlag = false; % Set to true if you want to save a video
hFig = figure(1); clf;
```




```
t = 25; % simulation time, [sec]
dt = 0.1; % time step, [sec]
K = t/dt; % number of frames = time of animation / time for each frame

if movieFlag
    v = VideoWriter('movie', 'MPEG-4');
    v.FrameRate = 1/dt;
    open(v);
end

% ______________Initial parameters________________

N = 20; % number of individuals
alpha = 270; % vision volume, [deg]
rr = 1; % zone of repulsion [units]
ro = 5; % zone of orientation [units]
ra = 15; % zone of attraction [units]
omega = 40; % max rotationa rate, [deg/sec]
s = 3; % speed of each individual, [units/sec]
```

```
myEps = 10e-6; % my small number

K = t/dt; % number of frames = time of animation / time for each frame

% INITIALIZE INDIVIDUALS

% ____random____
r = 5*rand(3,N) - 2.5;
v = rand(3,N) - 0.5;
%r(3,:) = 0;
%v(3,:) = 0;

% ____example 1____
%r(:,1) = [5; 0; 0]; r(:,2) = [-5; 0; 0];
%v(:,1) = [-1; 0; 0]; v(:,2) = [1; -1; 0];

%____example 2____
%r(:,1) = [5; 0; 0]; r(:,2) = [-5; 0; 0]; r(:,3) = [0; 0; 0]
%v(:,1) = [0; 1; 0]; v(:,2) = [0; 1; 0]; v(:,3) = [0; -1; 0]

for i = 1:N % normalize these vectors
    v(:,i) = v(:,i)/norm(v(:,i));
end

% matrices for information about the system, recorded each frame
r_group = zeros(3,K); % COM Position
v_group = zeros(3,K); % COM velocity
p_group = zeros(3,K); % COM linear momentum (equal to velocity if mass of system =
1)
h_group = zeros(3,K); % angular momentum about COM
r_inter = zeros(3,K); % distance of each fish from COM, used in calculating h_group

% ______________________Loop for each frame to be rendered___________________
for k = 1:K
    dir = zeros(3,N); % desired direction for each fish at the end of the frame

    for n = 1:N
       % variable reset
        dis = zeros(3,N); % distance list of each fish from a specific fish
        dirTemp = zeros(3,1); % temporary desired direction for a specific fish
        %dir = zeros(3,N);
        tempIndex = zeros(1,N); % temporary matrix for indices of important fish
        temp2 = zeros(3,N); % temporary matrix for location, velocity, etc. of
important fish
        temp3 = zeros(3,N); % ^^
        inrr = false; % booleans for if there are fish in zones
        inro = false; % ^^
        inra = false; % ^^
        angInit = 0; % initial angle calculated from velocity vector
        angTarg = 0; % target angle calcualted from dir vector
```

```
        dis1 = 0; % temporary variable for angular distance calculations
        dis2 = 0; % ^^
        angFinal = 0; % angle for new trajectory direction

        % __________________Determine zone for all other fish_________________
        for i = 1:N
            dis(:,i) = r(:,i) - r(:,n); % calculates distance between fish n and
all other fish i
            d = norm(dis(:,i));

            %_____blind spot______
            angle = acosd(dot(v(:,n),dis(:,i))/norm(v(:,n))/norm(dis(:,i)));    %
calculates angle between direction of fish n
                                                                                %
and position of fish i with respect to fish n

            if angle > 0.5*alpha % if the angle is outside the viewing range alpha,
it is labelled as 3 (ignore)
                tempIndex(1,i) = 3;
            end

            if d <= rr & tempIndex(1,i) ~= 3
                tempIndex(1,i) = 0; % categorize fish as in repulsion zone
            elseif d <= ro & tempIndex(1,i) ~= 3
                tempIndex(1,i) = 1; % categorize fish as in orientation zone
            elseif d <= ra & tempIndex(1,i) ~= 3
                tempIndex(1,i) = 2; %categorize fish as in attraction zone
            else
                tempIndex(1,i) = 3; % categorize fish as out of range
            end

            tempIndex(1,n) = 3; % ignore identical fish
        end

        %disp(tempIndex);

        % _____________zone of repulsion______________

        fishCount = 0;

        for i = 1:N
            if tempIndex(1, i) == 0
                inrr = true; % check if any fish at all are in the zone of repulsion
                fishCount = fishCount+1;
            end
        end

        if inrr
            for i = 1:N
                if tempIndex(1,i) == 0
```

```
                    temp2(:,i) = r(:,i); % if in zone of repulsion, puts position
in temp2 for later use
                end
            end

            target = [sum(temp2(1,:)/fishCount); sum(temp2(2,:)/fishCount); 0]; %
creates mean position of all relevant fish

            dir(:,n) = -(r(:,n) - target)/norm(r(:,n) - target); % determines
target dir for the specific fish

            continue; %cancels loop for particular fish, as all other fish are
outside repulsion zone
        end


        % ______________zones of orientation and
attraction___________________________

        % ____orientation____
        fishCount = 0;

        for i = 1:N
            if tempIndex(1, i) == 1
                inro = true; % check if any fish at all are in the zone of
orientation
                fishCount = fishCount+1;
            end
        end

        if inro
            for i = 1:N
                if tempIndex(1,i) == 1
                    temp2(:,i) = v(:,i); % if in zone of orientation, puts velocity
direction in temp2 for later use
                end
            end

            target = [sum(temp2(1,:)/fishCount); sum(temp2(2,:)/fishCount); 0]; %
creates mean direction of all relevant fish

            dirTemp = target/norm(target);

        end

        % _____attraction______
        fishCount = 0;
        for i = 1:N
            if tempIndex(1, i) == 2
```

```
                inra = true; % check if any fish at all are in the zone of
attraction
                fishCount = fishCount+1;
            end
        end

        if inra
            for i = 1:N
                if tempIndex(1,i) == 2
                    temp3(:,i) = r(:,i); % if in zone of attraction, puts position
in temp2 for later use
                end
            end

            %disp(temp3)

            target = [sum(temp3(1,:)/fishCount); sum(temp3(2,:)/fishCount); 0]; %
creates mean position of all relevant fish

            %disp(target)

            dir(:,n) = -(r(:,n) - target)/norm(r(:,n) - target); % determines
target dir for the specific fish
        end

        %disp(dir);


        %_____ combine_______

        if inro
            if inra
                dir(:,n) = (0.8*dir(:,n) + 0.2*dirTemp(:,1));
                dir(:,n) = dir(:,n)/norm(dir(:,n));
            else
                dir(:,n) = dirTemp(:,1);

            end
        end

        %disp(inrr)
        %disp(inro)
        %disp(inra)

    end

    %disp(dir);

    %__________________________________MOVEMENT______________________________
```

```
    for i = 1:N

        if dir(:,i) == [0;0;0]
            r(:,i) = r(:,i) + s*dt*v(:,i); % changes position with new velocity
            continue;
        end

        angFinal = 0;

        angInit = atan2d(v(2,i), v(1,i));
        if angInit < 0
            angInit = angInit + 360;
        end
        angTarg = atan2d(dir(2,i), dir(1,i)); % calculates signed angle of current
vector, and target vector
        %disp(angTarg)
        if angTarg < 0
            angTarg = angTarg + 360;
        end
        %disp(angInit);
        %disp(angTarg);

        dif1 = angTarg - angInit; % calculate both distances between angles
        dif2 = 360 - abs(dif1);

        %disp(dif1);
        %disp(dif2);


        if dif1 > 0 % if the first distance is positive, set dif1 to positive, dif2
to negative
            counterclockwise = dif1;
            clockwise = dif2;
        else
            counterclockwise = dif2; % if the first distance is negative, set
abs(dif1) to negative, dif2 to positive
            clockwise = abs(dif1);
        end

        %disp(counterclockwise);
        %disp(clockwise);

        if counterclockwise < clockwise % checks which direction is shortest,
applies movement in that direction
            if counterclockwise < omega*dt % makes sure fish will not overshoot turn
                angFinal = angTarg;
            else
                angFinal = angInit + omega*dt;
            end
        else
```

```
            if clockwise < omega*dt
                angFinal = angTarg;
            else
                angFinal = angInit - omega*dt;
            end
        end

        if angFinal > 360 % sets angFinal into proper bounds 0 - 360 degrees
            angFinal = angFinal - 360;
        elseif angFinal < 0
            angFinal = 360 + angFinal;
        end

        v(:,i) = [cosd(angFinal); sind(angFinal); 0]; % sets new velocity direction!

        r(:,i) = r(:,i) + s*dt*v(:,i); % changes position with new velocity
    end

    % _____________________________MOMENTUM_ETC_________________________

    % center of mass
    r_group(:,k) = [mean(r(1,:)); mean(r(2,:)); 0];

    % average velocity
    v_group(:,k) = s*[mean(v(1,:)); mean(v(2,:)); 0];

    % linear momentum
    p_group(:,k) = s/N*[sum(v(1,:)); sum(v(2,:)); 0];

    % angular momentum
    for j = 1:N
        r_inter(1,j) = r(1,j) - r_group(1,k);
        r_inter(2,j) = r(2,j) - r_group(2,k);

        h_group(:,k) = h_group(:,k) + s/N*cross(r_inter(:,j), v(:,j));
    end

    %_______Animation stuff__________
    figure(hFig);
    plot(r(1,:), r(2,:), 'o'); hold on;
    quiver(r(1,:), r(2,:), v(1,:), v(2,:), 0); hold off;
    titleStr = sprintf('t = %2.2f', k*dt);
    title(titleStr);
    axis equal;
    avgWindow = [sum(r(1,:))/N*[1,1], sum(r(2,:))/N*[1,1]];
    axis(window + avgWindow);
    drawnow;
    pause(0.25);

    if movieFlag
```

```
        frame = getframe(hFig);
        writeVideo(v, frame);
    end

end

if movieFlag
    close(v); % Close video file
end

% ___________________PLOTTING_____________________
tspan = dt : dt : t;

% center of mass
plot(tspan, r_group(1,:), tspan, r_group(2,:));
xlim([tspan(1) tspan(K)]); % sets the x axis limits to tspan
xlabel('Time (sec)') ;
ylabel('Center of mass');
legend('x','y');
clf

% average velocity/linear momentum
plot(tspan, v_group(1,:), tspan, v_group(2,:));
xlim([tspan(1) tspan(K)]); % sets the x axis limits to tspan
xlabel('Time (sec)') ;
ylabel('Velocity/linear momentum of C.O.M.');
legend('x','y');
clf

% angular momentum
plot(tspan, h_group(3,:), tspan, h_group(3,:));
xlim([tspan(1) tspan(K)]); % sets the x axis limits to tspan
xlabel('Time (sec)') ;
ylabel('Angular Momentum around C.O.M.');
clf
```