MECH 6V29 - HW 3

Author: Jonas Wagner

Date: 2022-03-22

```
clear;
clc;
close all;
```

Problem 6

The two different problems are:

1. Make sure that the swarm stays connected during rendezvous
2. Make sure that the robots don't run into each other

Proximity Disk Graph

```
Delta = 0.42; % interaction distance
network_no = 3; %% or 2 or 3

[X,n,N] = load_network(network_no, Delta);
% n = dimension of each robot (n>1)
% N = total number of robots
% X = n x N vector containing the initial robot positions
```

Some numerical integration parameters

```
dt = 0.003; % numerical steplength
Tf = 2; % final time

% Initial time
t = 0;
iter = 1;
```

Weight Function Controls

```
dist_goal = 0.1;
w_else = 2.2;
% p = 3;
% b = -tan(-pi/2 + pi/p);
%weightfcn = @(d) tan(-pi/2 + pi * d/Delta) + b;
weightfcn = @(d) (.5 .* (2 .* Delta - d)./(Delta - d).^2) * (d - dist_goal);
dist = @(xi, xj) sqrt( (xi - xj)' * (xi - xj));
weight = @(xi, xj) weightfcn(dist(xi, xj));

% H for hysterisys
epsilon = 0.05;
H = zeros(size(disk(X,N,Delta)));
dmin = inf;
```

Run Simulation and Plot

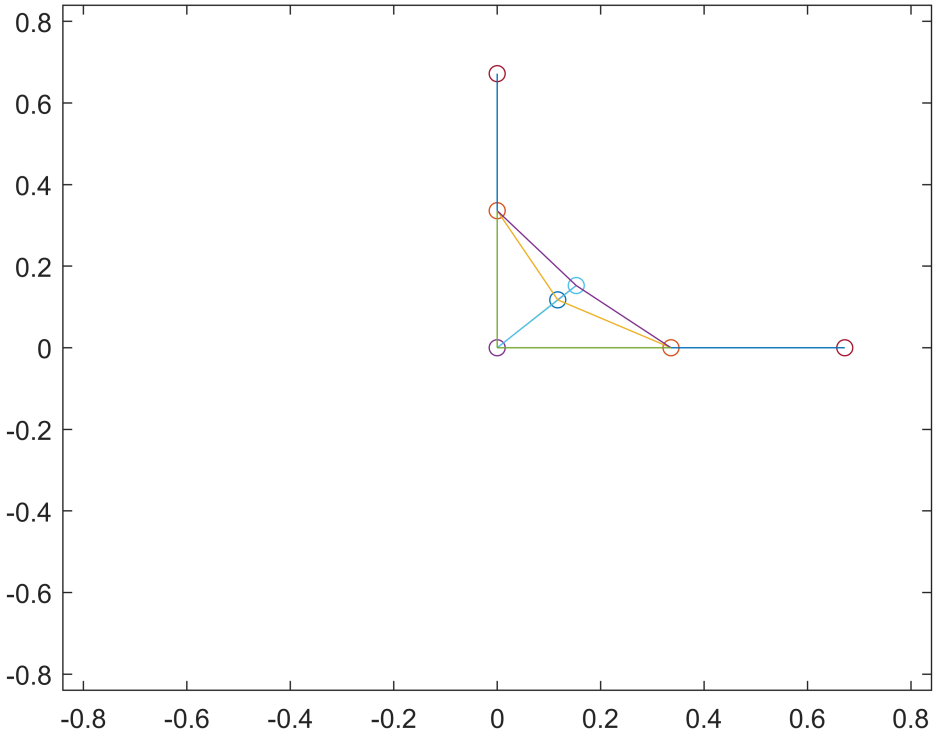
```
figure
while (t <= Tf)
    A = disk(X,N,Delta);

    DX=zeros(n,N); %% Here is where we store the derivatives
    for i=1:N
        for j=1:N
            if (A(i,j) == 1)
                dmin = min(dist(X(:,j), X(:,i)), dmin);
                if H(i, j) == 0     % this is a new edge
                    if dist(X(:,j), X(:,i)) < Delta - epsilon  % don't head towards
                        H(i, j) = 1;
                    end
                % Solution to avoid contacting each other
                else
                    if dist(X(:,j), X(:,i)) < dist_goal
                        H(i, j) = 1 / dist(X(:,j), X(:,i));
                    else
                        H(i,j) = 1;
                    end
                end
                w = weight(X(:,j), X(:,i)) * H(i, j);
                w = w + w_else * not(H(i,j)); % w = w_else if not below epsilon
                DX(:,i) = DX(:,i) + w .* (X(:,j) - X(:,i));   % The consensus equation
            end
        end
    end

    % Update the states using an Euler approximation
    for i=1:n
        X(:,i)=X(:,i)+dt.*DX(:,i);
    end

    % Update time
    t=t+dt;
    % Plot the solution every 10 iterations
    if (mod(iter,10)==0)
        plotsol(X,N,A,Delta);
    end

    iter = iter+1;
end
```




```
dmin
```

```
dmin = 0.0493
```