Université de Technologie
de Compiègne

Master Ingénierie des Systèmes Complexes ARS

ARS1 Project Report

Rendez-vous Problem + Target

Réaliser par :

Yassine Mathlouthi
Fadi Ben Ali
Adnane Salili

Encadré par :
Dr.Reine Talj Kfoury

2024-2025

Contents

List of Figures

1 Introduction

In the realm of autonomous systems, the coordination of multiple robots presents a fascinating and complex challenge that spans across various domains, from industrial automation to autonomous vehicular systems. Our project, titled "Rendez-vous Problem + target", focuses on the orchestrated movement of six autonomous robots, aiming to explore the dynamics of their interactions and convergence to a common meeting point. This project tests the fundamental principles of robotic communication and control through consensus-based control laws.

The objectives of this project are:

1. **Meeting Convergence**: Facilitate a rendez-vous where all robots converge to a single point through direct control of their speeds.
2. **Control Strategy Development**: Develop and implement a consensus-based control law that ensures all agents can meet on a common point.
3. **Graphical Representation and Simplification**: Represent the communication among robots as a graph and propose a simplified version that maintains the system's functionality.
4. **Simulation and Analysis**: Utilize Matlab/Simulink to simulate the control system, analyze the impact of different controller gains, and verify the convergence through graphical plots of the robots' trajectories.
5. **Control Strategy Development**: Develop and implement a consensus-based control law that ensures all agents can meet on a common point then continue to travel together on the curve $y = 2.x^3$.

This project is set against the backdrop of initial conditions where robots are placed at specific coordinates, reflecting a diverse range of starting points. The rendez-vous problem is not only a test of algorithmic precision but also a showcase of the interplay between individual robot behaviors and the overarching system dynamics. Through this project, we aim to elucidate the effectiveness of distributed control laws in multi-agent systems and enhance our understanding of collective robotic behaviors in complex environments.

2 Graphical Model of the system

2.1 Initial Conditions

In our project we suppose the case of six robots, where we can control directly their speed. V= [Robot1, Robot2, Robot3, Robot4, Robot5, Robot6] is the nonempty finite node set of our system, where the initial positions of each robots are Robot1(-7,4), Robot2(-4,-3), Robot3(7,-2), Robot4(9,5), Robot5(0,-6) and Robot6(5,10).

In order to exchange information among all the robots, we model by the simplest directed graph, where vertices will represent robots and edges are the information exchange links among robots. Let $E \subseteq V \times V$ be the edge set, where an edge $(i, j) \in E$ means that the Robot j can get updates from robot i.

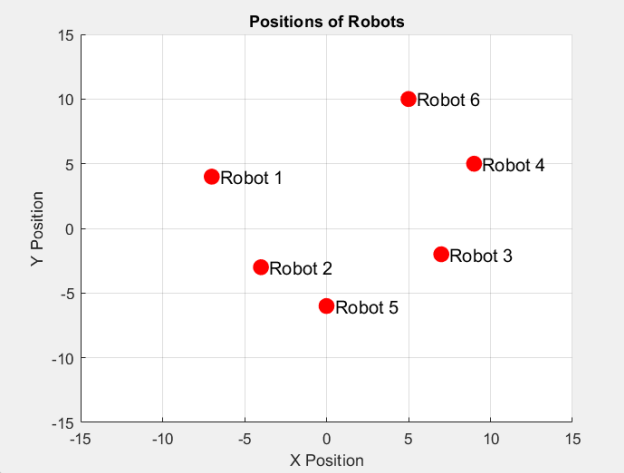


Figure 1: Intitial positions of the Robots

2.2 Fully garph connected

In our case, we will have a strongly connected graph, because all robots communicate with each other.

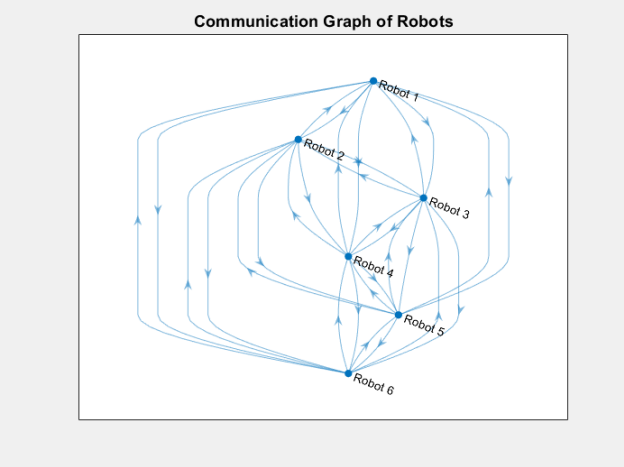


Figure 2: Communication Graph of the Robots 1

The elements of the adjacency (communication) matrix of our system $G = [g_{ij}] \in R^{n \times n}$ will be $g_{ij} = 1$ if robot i receives a communication signal from robot j and 0 otherwise.

$$G1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

2.3 Simplified Graph

To maintaine essential communication information, we reduced the connections between the robots.Each robot communicates with his nearest neighbor.

The following figure represents the new system graph: The elements of the adjacency (communication)

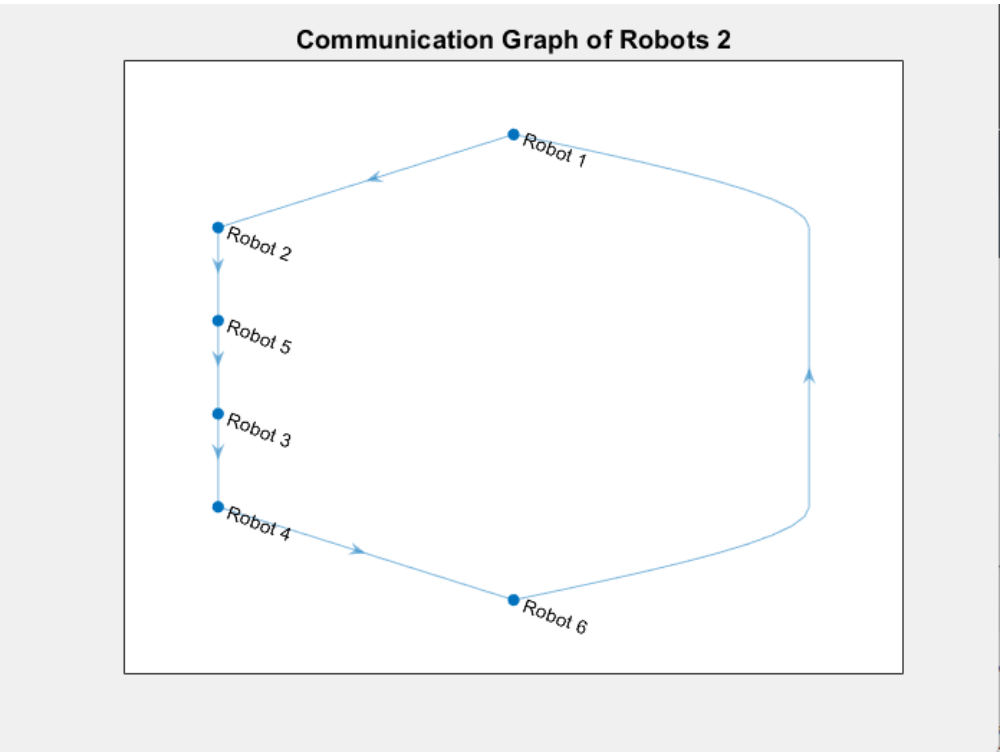


Figure 3: Simplified Communication Graph

matrix of our system $G = [g_{ij}] \in R^{n \times n}$ will be $g_{ij} = 1$ if robot i receives a communication signal from robot j and 0 otherwise.

$$G2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.4 Robot1 is the Leader

Now robot 1 the the leader and each other robot communicates with his nearest neighbor.
The elements of the adjacency (communication) matrix of our system $G = [g_{ij}] \in R^{n \times n}$ will be $g_{ij} = 1$ if robot i receives a communication signal from robot j and 0 otherwise.

$$G3 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The following figure represents the new system graph:

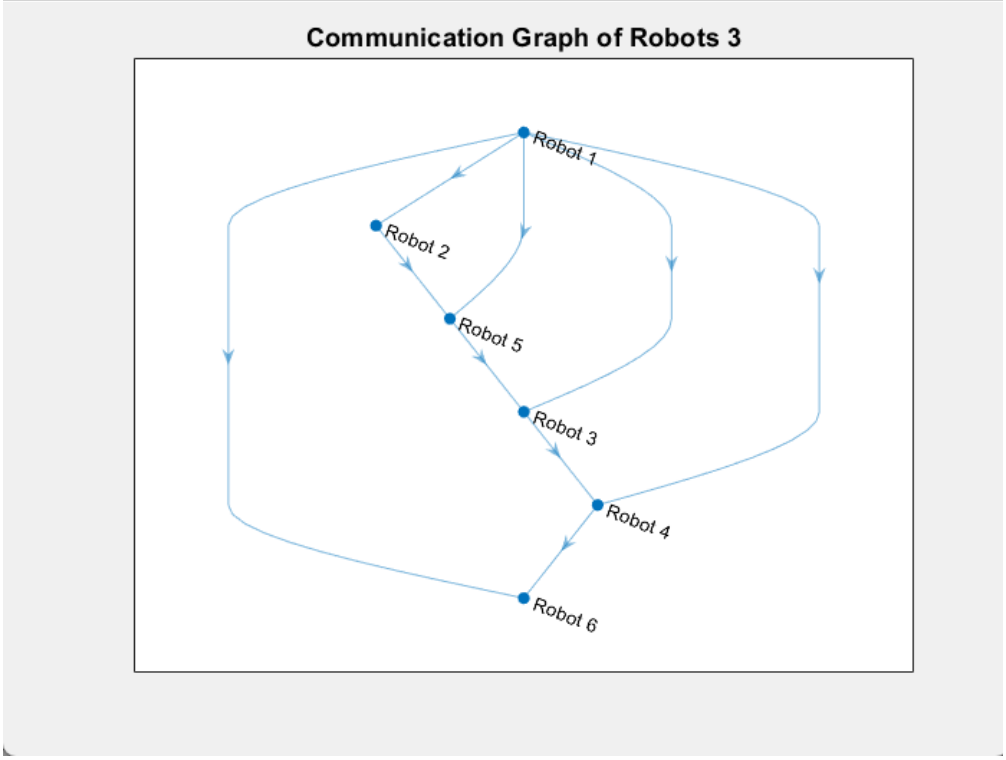


Figure 4: The robot1 is the leader

In the next section, we will present the theory behind collaborative control.

3 Consensus-based Control Law

3.1 Model of a single robot

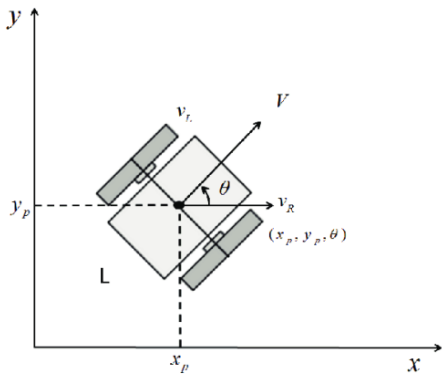


Figure 5: Kinematic Model of a Robot

The kinematic equations of a single robot can be described as follows:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases} \tag{1}$$

However, we need to choose a point away from the center of gravity of the robot to satisfy the holonomic conditions:

$$\begin{cases} x_h = x + L\cos(\theta) \\ y_h = y + L\sin(\theta) \end{cases} \tag{2}$$

Where:

- L is the distance from the center of gravity of the robot and the holonomic point

Derivation of the Position Equations :

Taking the derivative on both sides of each equation in (2) and substituting from (1), we get:

$$\begin{bmatrix} \dot{x}_h \\ \dot{y}_h \end{bmatrix} = \begin{bmatrix} \cos\theta & -L\sin\theta \\ \sin\theta & L\cos\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3}$$

Note that:

$$|R| = L > 0 \implies R \text{ is invertible} \implies R^{-1} \text{ exists.} \tag{4}$$

Hence we can define new input U such that

$$V = R^{-1}U \implies \tag{5}$$

$$\begin{bmatrix} \dot{x}_h \\ \dot{y}_h \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{6}$$

3.2 Control Law

3.2.1 Control Law to let all the robots meet

In the cooperative control consensus, robots will share their information about positions or speeds with each other. The topology of our system allows at any time each robot to access information of another robot and to get updates from them. Each robot will regulate its position with respect to its neighbors.
The Consensus-Based Control Law will specify the exchange of information within the system, between a robot and all its nearby neighbors, and will model it with differential equations. The control input is described as below:

$$\begin{aligned} U_{x_i} &= -\sum_{j=1}^{n} g_{ij}K_{ij} \cdot (x_{h_i} - x_{h_j}), \\ U_{y_i} &= -\sum_{j=1}^{n} g_{ij}K_{ij} \cdot (y_{h_i} - y_{h_j}). \end{aligned} \tag{7}$$

Where :

- x_{h_i}: The state variable representing the position of robot i at any time t.
- x_{h_j}: The state variable representing the position of robot j at any time t.
- y_{h_i}: The state variable representing the position of robot i at any time t.
- y_{h_j}: The state variable representing the position of robot j at any time t.
- K_{ij}: Gain correlating communication from robot i to robot j.
- g_{ij}: Entries of the adjacency matrix.

3.2.2 Control Law to let all the robots meet at a specif point

$$
\begin{aligned}
U_{x_i} &= -K_{d_i} \cdot (x_{h_i} - x_{f_i}) - \sum_{j=1}^{n} g_{ij} K_{ij} \cdot ((x_{h_i} - x_{h_j}) - (x_{f_i} - x_{f_j})), \\
U_{y_i} &= -K_{d_i} \cdot (y_{h_i} - y_{f_i}) - \sum_{j=1}^{n} g_{ij} K_{ij} \cdot ((y_{h_i} - y_{h_j}) - (y_{f_i} - y_{f_j})).
\end{aligned}
\tag{8}
$$

Where :

- x_{h_i}: The state variable representing the position of robot i at any time t.
- x_{h_j}: The state variable representing the position of robot j at any time t.
- y_{h_i}: The state variable representing the position of robot i at any time t.
- y_{h_j}: The state variable representing the position of robot j at any time t.
- x_{f_i}: Constant representing the final target tracked by the robot i.
- x_{f_j}: Constant representing the final target tracked by the robot j.
- y_{f_i}: Constant representing the final target tracked by the robot i.
- y_{f_j}: Constant representing the final target tracked by the robot j.
- K_{d_i}: Gain associated to track final target of robot i.
- K_{ij}: Gain correlating communication from robot i to robot j.
- g_{ij}: Entries of the adjacency matrix.

3.2.3 Control Law to let all the robots follow the curve $'y = 2x^{3\prime}$

$$
\begin{aligned}
&U_{x_i} = 0.2, \\
&U_{y_i} = -K_{d_i} \cdot (y_{h_i} - \mathrm{y} = 2\mathrm{x}^3) \text{ - } \textstyle\sum_{j=1}^{n} g_{ij} K_{ij} \cdot ((y_{h_i} - y_{h_j})),
\end{aligned}
\tag{9}
$$

Where :

- y_{h_i}: The state variable representing the position of robot i at any time t.
- y_{h_j}: The state variable representing the position of robot j at any time t.
- K_{d_i}: Gain associated to track final target of robot i.
- K_{ij}: Gain correlating communication from robot i to robot j.
- g_{ij}: Entries of the adjacency matrix.

3.3 Convergence Proof

We have, $\dot{X} = AX$, where X is the state variables vector defined by:

$$
X = \begin{bmatrix} x_{h1} & y_{h1} & x_{h2} & y_{h2} & x_{h3} & y_{h3} & x_{h4} & y_{h4} & x_{h5} & y_{h5} & x_{h6} & y_{h6} \end{bmatrix}^T
$$

$$
\begin{bmatrix} \dot{x}_{h1} \\ \dot{y}_{h1} \\ \dot{x}_{h2} \\ \dot{y}_{h2} \\ \dot{x}_{h3} \\ \dot{y}_{h3} \\ \dot{x}_{h4} \\ \dot{y}_{h4} \\ \dot{x}_{h5} \\ \dot{y}_{h5} \\ \dot{x}_{h6} \\ \dot{y}_{h6} \end{bmatrix} = \begin{bmatrix} K_T & 0 & K & 0 & K & 0 & K & 0 & K & 0 & K & 0 \\ 0 & K_T & 0 & K & 0 & K & 0 & K & 0 & K & 0 & K \\ K & 0 & K_T & 0 & K & 0 & K & 0 & K & 0 & K & 0 \\ 0 & K & 0 & K_T & 0 & K & 0 & K & 0 & K & 0 & K \\ K & 0 & K & 0 & K_T & 0 & K & 0 & K & 0 & K & 0 \\ 0 & K & 0 & K & 0 & K_T & 0 & K & 0 & K & 0 & K \\ K & 0 & K & 0 & K & 0 & K_T & 0 & K & 0 & K & 0 \\ 0 & K & 0 & K & 0 & K & 0 & K_T & 0 & K & 0 & K \\ K & 0 & K & 0 & K & 0 & K & 0 & K_T & 0 & K & 0 \\ 0 & K & 0 & K & 0 & K & 0 & K & 0 & K_T & 0 & K \\ K & 0 & K & 0 & K & 0 & K & 0 & K & 0 & K_T & 0 \\ 0 & K & 0 & K & 0 & K & 0 & K & 0 & K & 0 & K_T \end{bmatrix} \begin{bmatrix} x_{h1} \\ y_{h1} \\ x_{h2} \\ y_{h2} \\ x_{h3} \\ y_{h3} \\ x_{h4} \\ y_{h4} \\ x_{h5} \\ y_{h5} \\ x_{h6} \\ y_{h6} \end{bmatrix}
$$

where $K_T = -5K_{ij} - K_d$. We used the same gain K between robots for communication, and same target gain K_d, for each robot.
Since $K_T < 0$ for positive K_{ij} and K_d, the diagonal values of the matrix A of the system are non-positive, so the system is stable.
where A is a constant matrix. This system of first order differential equations has the solution:

$$X(t) = e^{At}X(0)$$

Knowing that A has negative eigenvalues:

$$\lim_{t\to\infty} e^{At} = 0_{[n\times n]}$$

where n is the dimension of A. Therefore, as $t \to \infty$, $X(t) \to 0_{[n\times 1]}$, which proves that all robots' positions converge to the origin.

4 Implementation using MATLAB/SIMULINK

4.1 Implementation using MATLAB

To implement the controlled system described above, we have developed a function in `MATLAB`. This function takes input from the file `"ars1_init.m"`, which provides the initial positions of the robots (with $x_h(i)$ and $y_h(i)$ assumed to be their initial coordinates), the position of the target, the communication gain K_{ij} (identical for all robots), the target gain K_d (also the same for all robots), and the adjacency matrix G.

The first function will generate plots of the robots' trajectories and the variation of the distance between each robot and the target over time, illustrating the system's response speed. The code of the function is provided below:

```
function simulate_robots(n, time, dt, x_h, y_h, x_f, y_f, g1, K_d, K_ij)
 % Initialize positions
 x = zeros(n, length(time));
 y = zeros(n, length(time));
 distances = zeros(n, length(time));

 % Set initial positions
 x(:, 1) = x_h';
 y(:, 1) = y_h';
 A = zeros(2*n, 2*n);
 for i = 1:n
     for j = 1:n
         if i ~= j
             A(2*i-1, 2*j-1) = g1(i, j) * K_ij;  % X interactions
             A(2*i, 2*j) = g1(i, j) * K_ij;      % Y interactions
         end
     end
     A(2*i-1, 2*i-1) = -K_d-5*K_ij;  % Self-damping X
     A(2*i, 2*i) = -K_d-5*K_ij;      % Self-damping Y
 end
 disp(A)
 % Run the simulation (Euler method for simplicity)
 for t = 2:length(time)
     for i = 1:n
         % Control law
         U_xi = -K_d * (x(i, t-1) - x_f(i)); % traction to the target
         U_yi = -K_d * (y(i, t-1) - y_f(i));

         % the influence of other robots
         for j = 1:n
             if i ~= j
                 U_xi = U_xi - g1(i, j) * K_ij * ((x(i, t-1) - x(j, t-1)) - (x_f(i) - x_f
(j)));
```

```
                    U_yi = U_yi  - g1(i, j) * K_ij * ((y(i, t-1) - y(j, t-1)) - (y_f(i) -
    y_f(j)));


                end
            end
            disp(U_xi)
            disp(U_yi)

            % Update positions (simple Euler integration)
            x(i, t) = x(i, t-1) + U_xi * dt;
            y(i, t) = y(i, t-1) + U_yi * dt;

            % Calculate distance from the target
            distances(i, t) = sqrt((x(i, t)^2 + y(i, t)^2) - ((x_f(i)^2 + y_f(i)^2) ));
        end
    end
disp(A)
eig(A)
    % Plot the trajectories of the robots
    figure;
    hold on;
    axis([-15 15 -15 15]);
    for i = 1:n
        plot(x(i, 1), y(i, 1), 'ro'); % Initial position
        text(x(i, 1) + 0.5, y(i, 1), sprintf('Robot %d', i), 'FontSize', 12);
        plot(x_f(i), y_f(i), 'gx', 'LineWidth', 2);
        text(x_f(i) - 4, y_f(i), sprintf('Target'), 'FontSize', 12);
        plot(x(i, :), y(i, :), 'LineWidth', 2); % Trajectory
    end

    xlabel('X Position');
    ylabel('Y Position');
    title(sprintf('Trajectories of Robots for K_d= %d and K_ij= %d', K_d, K_ij));
    grid on;
    hold off;

    % Plot the distances from the origin vs time
    figure;
    hold on;
    for i = 1:n
        plot(time, distances(i, :), 'LineWidth', 1);
    end
    xlabel('Time (s)');
    ylabel('Distance to Target');
    title('Distances from Robots to the target vs. Time');
    grid on;
    hold off;
end
```

The second function will generate plots of the robots' trajectories and the variation of the distance between each robot and the target over time, and when all the robots reach the target it will plot the trajectories following the curve "$y = 2x^3$" illustrating the system's response speed. The code of the function is provided below:

```
   function simulate_robots2(n, time, dt, x_h, y_h, x_f, y_f, g1, K_d, K_ij)
    % Initialize positions
    x = zeros(n, length(time));
    y = zeros(n, length(time));
    distances = zeros(n, length(time));
    has_met = false;  % Indicator if all robots met at the meeting point

    % Set initial positions
    x(:, 1) = x_h';
    y(:, 1) = y_h';

```

```
    % Initialize control variables for the curve phase
    curve_start_time = 0;  % Time when curve following starts
    curve_speed = 0.2;  % Speed along x-axis for curve following

    % Run the simulation (Euler method for simplicity)
    for t = 2:length(time)
        for i = 1:n
            if ~has_met
                % Control law for moving towards the meeting point
                U_xi = -K_d * (x(i, t-1) - x_f(i));
                U_yi = -K_d * (y(i, t-1) - y_f(i));
            else
                % Control law for moving along the curve y = 2x^3
                if t == curve_start_time
                    x(i, t) = x_f(i);  % Synchronize x positions
                    y(i, t) = y_f(i);  % Synchronize y positions
                end
                U_xi = curve_speed;
                U_yi = -K_d * (y(i, t-1) - 2 * x(i, t-1)^3);
            end

            % Sum of the influence of other robots (if necessary)
            for j = 1:n
                if i ~= j
                    U_xi = U_xi - g1(i, j) * K_ij * (x(i, t-1) - x(j, t-1));
                    U_yi = U_yi - g1(i, j) * K_ij * (y(i, t-1) - y(j, t-1));
                end
            end

            % Update positions (simple Euler integration)
            x(i, t) = x(i, t-1) + U_xi * dt;
            y(i, t) = y(i, t-1) + U_yi * dt;
        end

        % Check if all robots have met at the meeting point
        if ~has_met && all(sqrt((x(:, t) - x_f(i)).^2 + (y(:, t) - y_f(i)).^2) < 0.0001)
            has_met = true;  % All robots have met
            curve_start_time = t;  % Set the time when curve following starts
        end
    end

    % Plotting results
    figure;
    hold on;
    axis equal;
    grid on;
    for i = 1:n
        plot(x(i, :), y(i, :), 'LineWidth', 2); % Trajectory
        plot(x(i, 1), y(i, 1), 'ro'); % Initial position
        text(x(i, 1) + 0.5, y(i, 1), sprintf('Robot %d', i), 'FontSize', 12);
        plot(x_f(i), y_f(i), 'gx', 'LineWidth', 2); % Meeting point

    end
    title('Robot trajectories and meeting point');
    xlabel('X Position');
    ylabel('Y Position');
    legend('Trajectories', 'Start Positions','Meeting point', 'Location', 'best');
    hold off;
end
```

4.2 Implementation using SIMULINK

We also implement the control laws in simulink but the visualisation of the trajectories is better with the function plot of matlab
The following figure represents the new system graph:

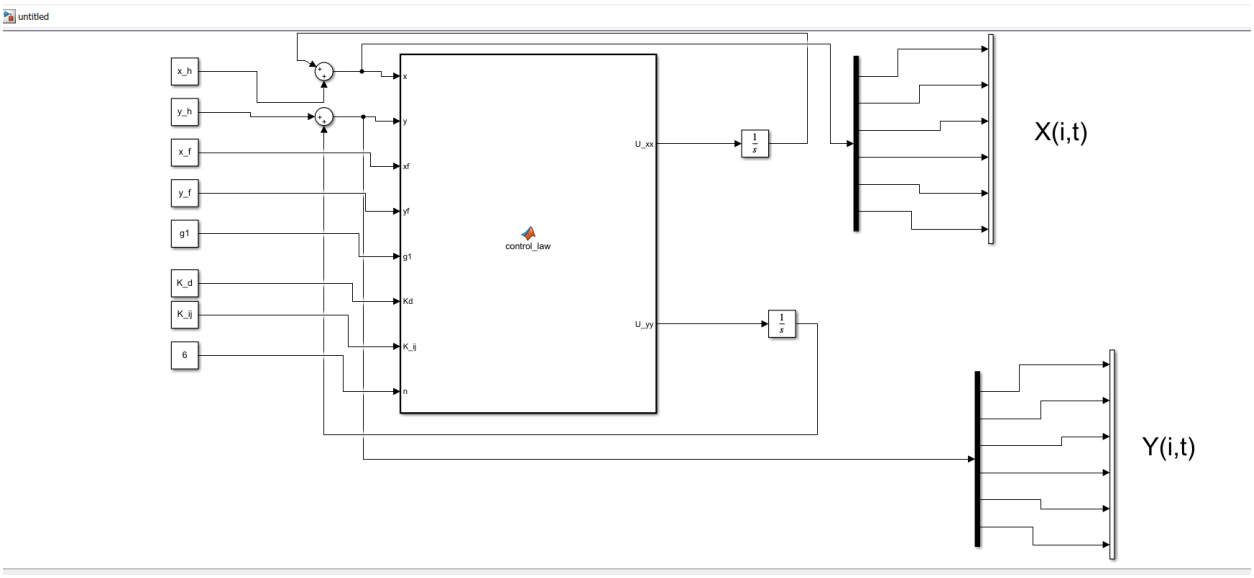


Figure 6: Implementation in SIMULINK

5 Results Analysis

5.1 Fully connected garph

First we have the result"All robots meet": In this casee the robots attempt to get closer, and these successive adjustments ultimately lead to all the robots converging toward a common central point.

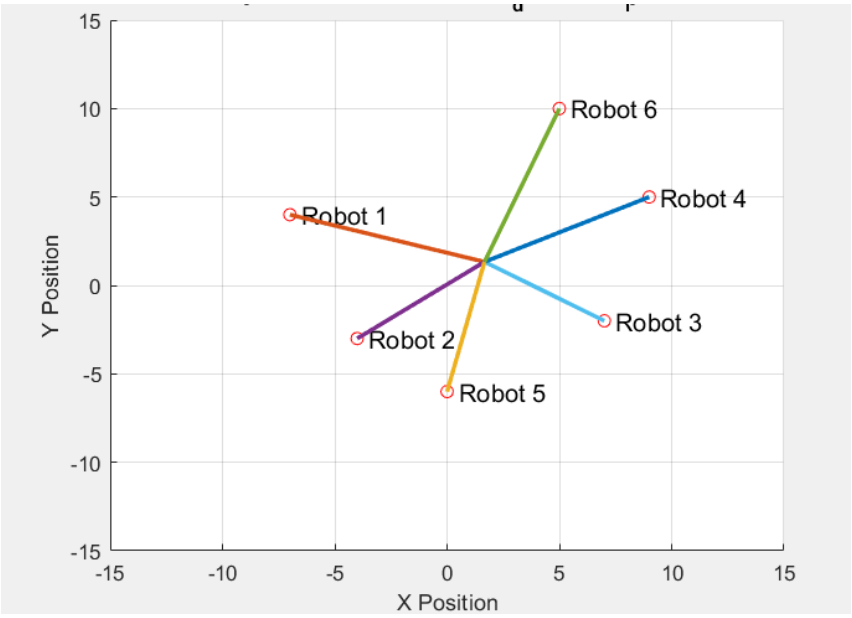
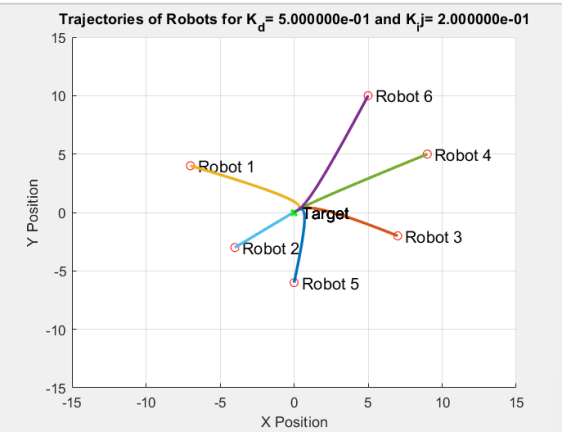
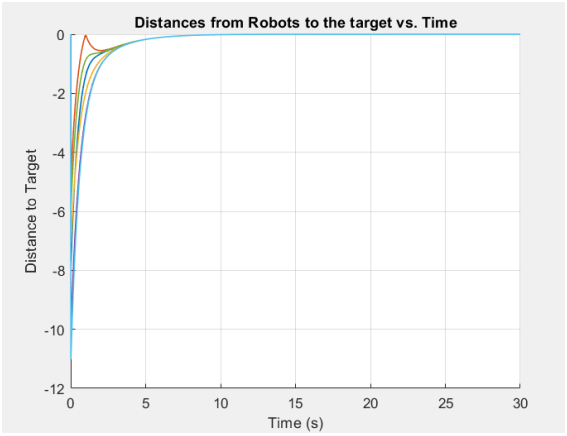


Figure 7: All robots meet

Now all robot will meet at the target point(0,0) with deffirent values of K_d and K_ij:

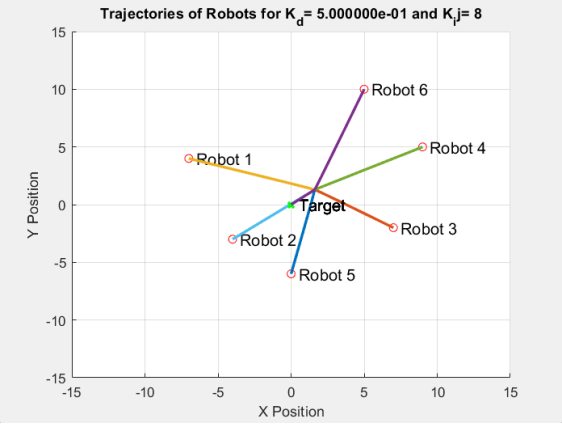


(a) Trajectories for $K_d = 0.5$ and $K_ij = 0.2$

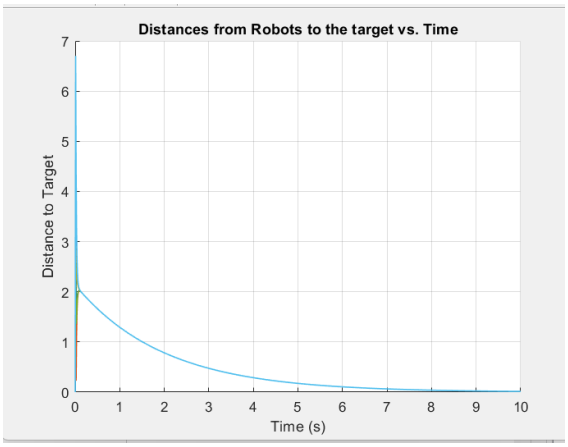


(b) Distances from Robots to Origin vs. Time $K_d = 0.5$ and $K_ij = 0.2$

Figure 8: Simulation for $K_d = 0.5$ and $K_ij = 0.2$

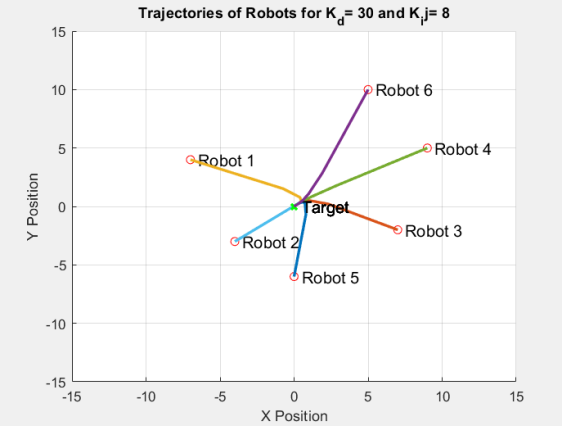


(a) Trajectories for $K_d = 0.5$ and $K_ij = 8$

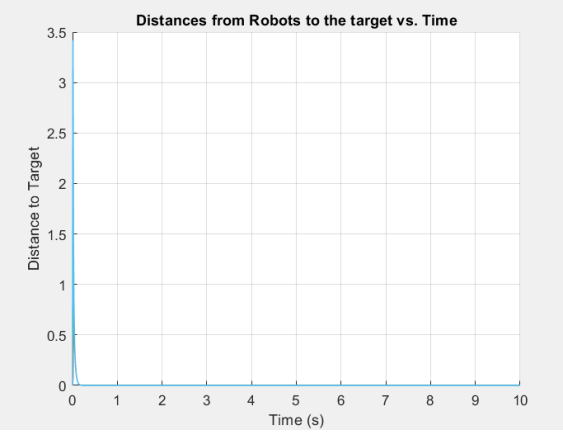


(b) Distances from Robots to Origin vs. Time $K_d = 0.5$ and $K_ij = 8$

Figure 9: Simulation for $K_d = 0.5$ and $K_ij = 8$



(a) Trajectories for $K_d = 30$ and $K_ij = 8$



(b) Distances from Robots to Origin vs. Time $K_d = 30$ and $K_ij = 8$

Figure 10: Simulation for $K_d = 30$ and $K_ij = 8$

As the plots show, higher gain values lead to faster response times. However, adjusting the gains between the robots, as opposed to the gains aimed at reaching the target, produces different transient behavior. Larger values of K_ij cause the robots to prioritize meeting each other over reaching the final destination. Similarly, increasing K_d shifts the focus towards reaching the final destination.

5.2 Simplified graph

First we have the result "All robots meet": In this case, the robots are trying to meet using a simplified graph, where each robot communicates only with its closest neighbor. The observed spiral trajectories result from the constant adjustments each robot makes in relation to its neighbor, creating spiral movements. Each pair of robots attempts to get closer, and these successive adjustments ultimately lead to all the robots converging toward a common central point.

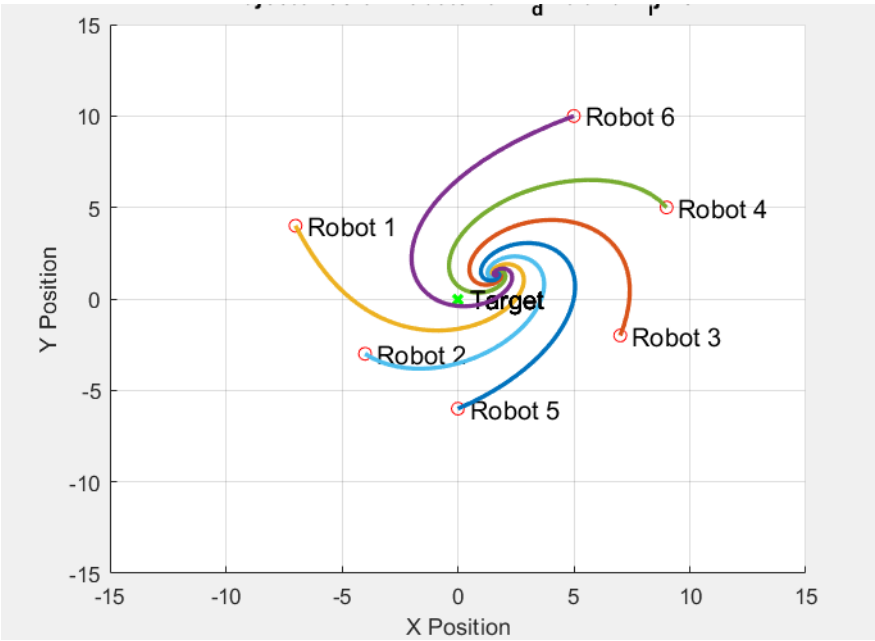
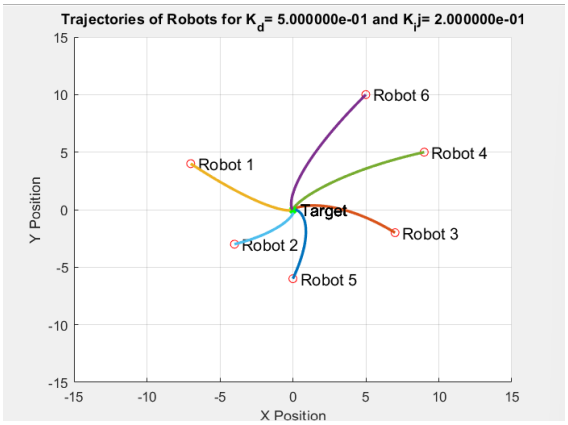
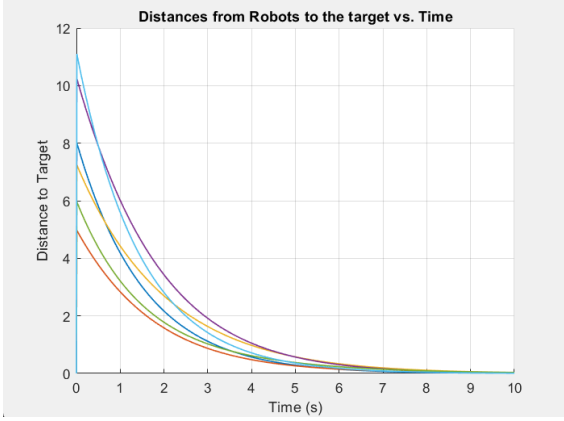


Figure 11: All robots meet for the simplified graph

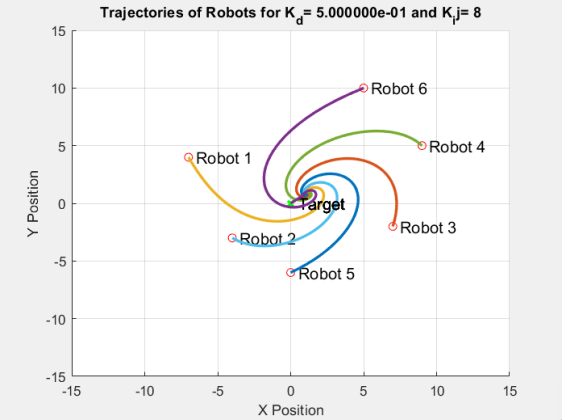


(a) Trajectories for $K_d = 0.5$ and $K_ij = 0.2$

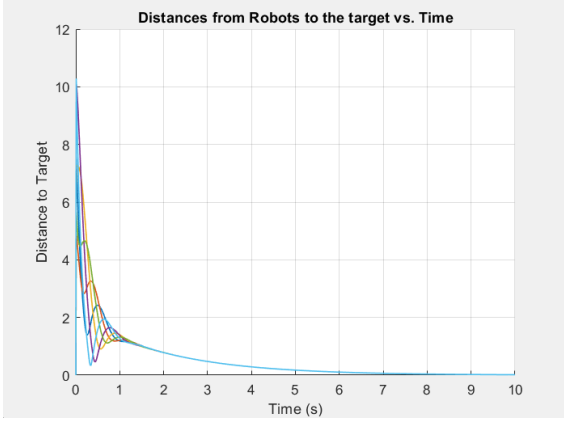


(b) Distances from Robots to Origin vs. Time $K_d = 0.5$ and $K_ij = 0.2$

Figure 12: Simulation for $K_d = 0.5$ and $K_ij = 0.2$

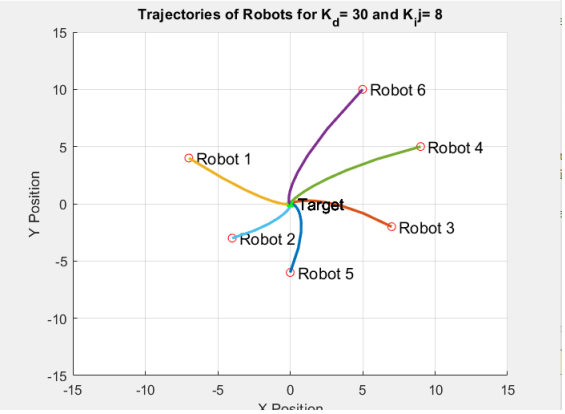


(a) Trajectories for $K_d = 0.5$ and $K_ij = 8$

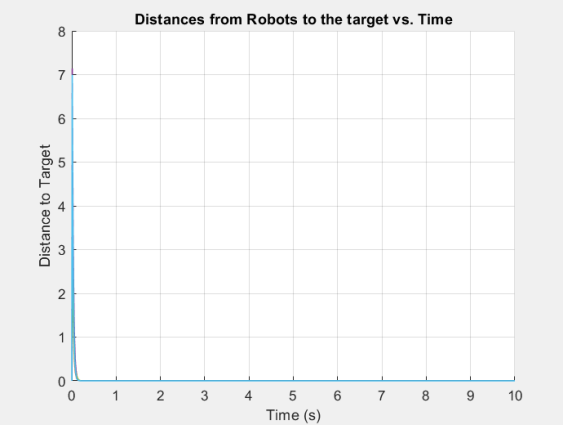


(b) Distances from Robots to Origin vs. Time $K_d = 0.5$ and $K_ij = 8$

Figure 13: Simulation for $K_d = 0.5$ and $K_ij = 8$



(a) Trajectories for $K_d = 30$ and $K_ij = 8$



(b) Distances from Robots to Origin vs. Time $K_d = 30$ and $K_ij = 8$

Figure 14: Simulation for $K_d = 30$ and $K_ij = 8$

Like the others configurations, higher gain values lead to faster response times. However, adjusting the gains between the robots, as opposed to the gains aimed at reaching the target.
Larger values of K_ij cause the robots to prioritize meeting each other over reaching the final destination. Similarly, increasing K_d shifts the focus towards reaching the final destination.

5.3 All the robots meet at the position of the first robot

We used G3 graph where the robot 1 is the leader: and now all the robots will meet at the position of the robot1

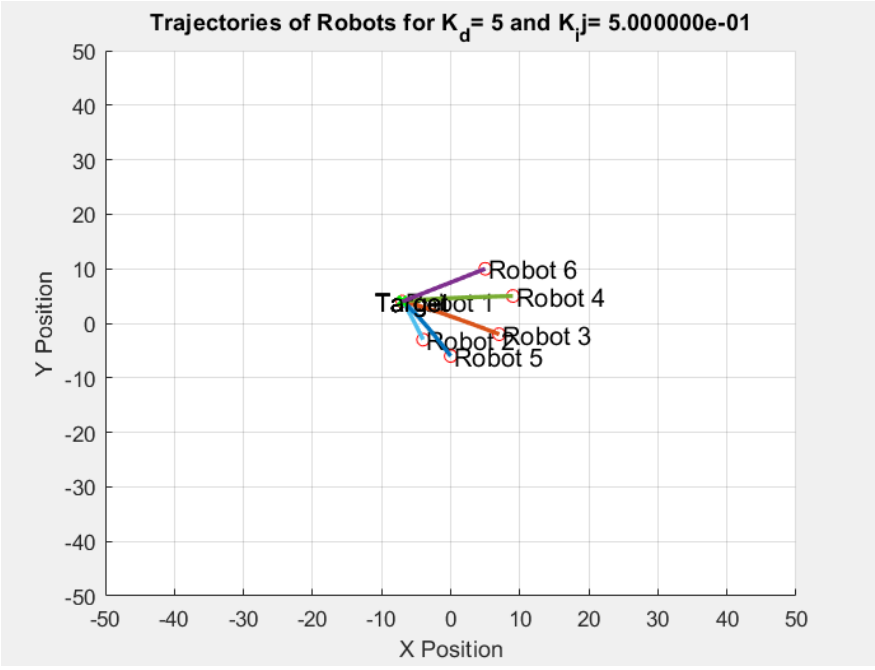


Figure 15: All the robots meet at the position of the first robot

5.4 All the robots meet at the meeting point (0,0) and continue to travel together on the curve '$y = 2x^3$'

We used the adjcency matrix G2 and the control low to reach the meeting point then we change the control law to follwing the curve '$y = 2x^3$'

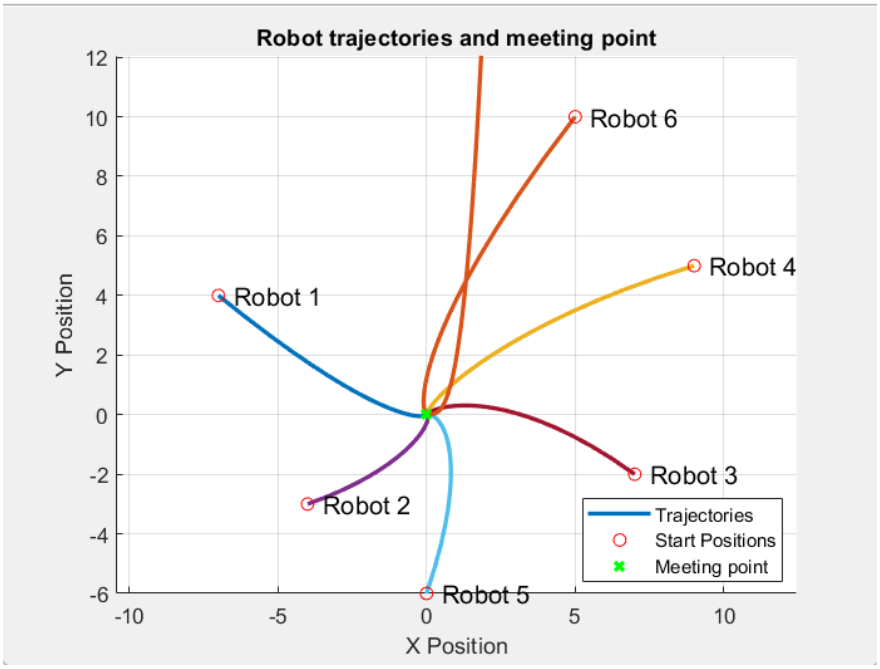


Figure 16: All robots meet and follow the curve

6 Conclusion

In this project, we successfully explored the rendez-vous problem for a group of six robots. The objective was to design control strategies that enable the robots to meet at a specific location or follow a common trajectory. Through simulations in MATLAB/Simulink, we implemented and tested various consensus-based control laws, starting with a fully connected communication graph where all robots interacted with each other. We analyzed the results by varying the control gains and observed how the robots' trajectories converged toward the meeting point. We further simplified the communication graph, reducing the number

of connections while maintaining the convergence property. The simplified graph led to interesting spiral trajectories due to the reduced interaction between neighboring robots, yet the system still converged to a common point. Additionally, we demonstrated that with a proper selection of control gains, the system's stability is ensured, as proven by the negative eigenvalues of the system's state matrix.

Through these simulations, we gained valuable insights into the effects of communication topology and control gains on the robots' behavior. The project shows the robustness of consensus-based control strategies for multi-agent systems and their ability to solve rendez-vous problems efficiently.

7 References

- **ARS1 course**