Université de Technologie de Compiègne

Advanced Method of Control (ARS1)

Rendez-vous Problem with target

Presented to:

Dr.Reine Talj Kfoury

Done by:

Ahmad Shour
Hasan Kasem
Hussein Lezzaik

The aim of this project is to design a Conesus based control of 5 robots, starting from initial position to a desired location.

Contents

List of Figures

1 Introduction:

Multi-agent systems become a wide field of study and research. During the past decades, researches are heavily done on the control of multi-agent systems. This refers to their importance. A group of agents can achieve more tasks than having one agent and sometimes they achieve assignments that can never be done by a single agent.

This is done through their cooperative way in work, and this makes them faster, cheaper and more reliable than one agent. The domain of multi-agent systems has wide range of applications; starting from the manipulation and transport of objects, exploration, navigation, localization and motion coordination. The latter is a navigation based on a specific strategy for a set of agents..

Consensus is an important problem in cooperative control, where several agents have to be synchronized to a common value. In the case of multi-robots systems, it can be interesting for the robots to achieve a consensus on positions, like a rendez-vous point , to reach an objective at the same instant. After studying the concepts of Systems of Systems in the subject of ARS1, we've choosen to implement the " Rendez-Vous with target case ", where we assume the case of five robots with which we can directly control their speeds.

The purpose of our project is to propose a consensus-based control law that lets the agents meet in the specified location (0,0), study the behavior of our system and simplify our graphical model.

At first we plot the graph of the system, then we propose a consensus-based control law that let the agents meet in the specified location. After that we implement the control system in MATLAB Simulink and analyze the results function of the controllers gains. We then proposed another simplified graph with less connections, adapted the proposed controller and analyzed the simulation results. Finally, we try to propose a controller that lets the robots meet, then continue to travel together with respect to the 'x-axis' direction at constant speed.

2 Graphical Model of the system:

2.1 Initial Positions :

In our project we suppose the case of five robots, where we can control directly their speed.

V= [Robot1, Robot2, Robot3, Robot4, Robot5] is the nonempty finite node set of our system, where the initial positions of each robots are R1(0,8), R2(-2,-4), R3(3,-1), R4(6,4) and R5(-4,7). Agents have to meet in the specific location (0, 0).

We suppose that all the robots can communicate between themselves.

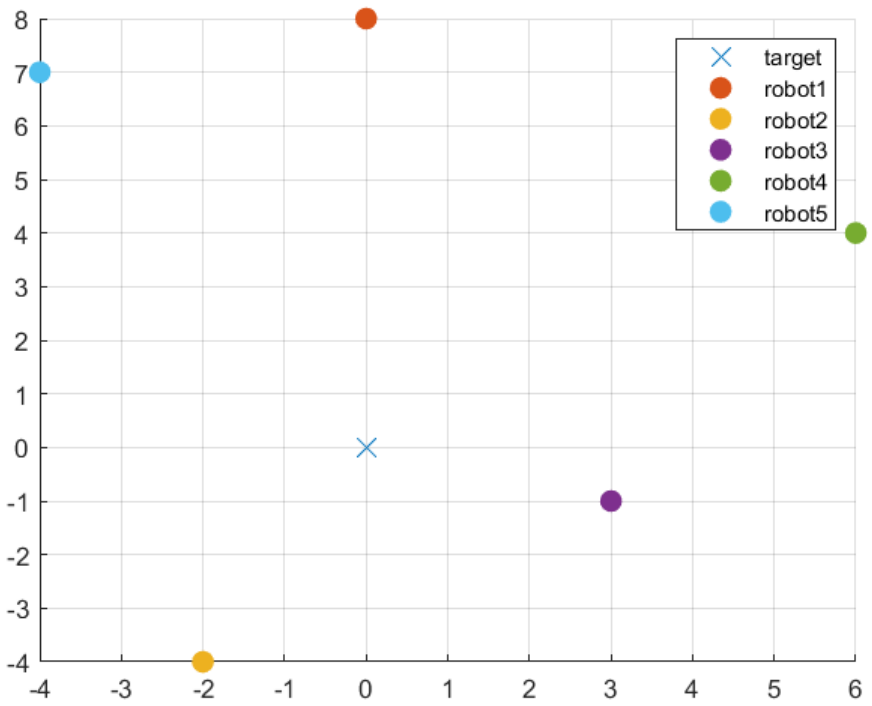

Figure 1: Initial Positions

2.2 Graph of our System:

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.

E $\subseteq$ [V $\times$ V] is the edge set, where edge (i, j) $\in$ E means that the Robot j can get updates from robot i, but for the Robot i it is not permissible.

In our case, we will have a strongly connected graph, because all robots communicate

between themselves.

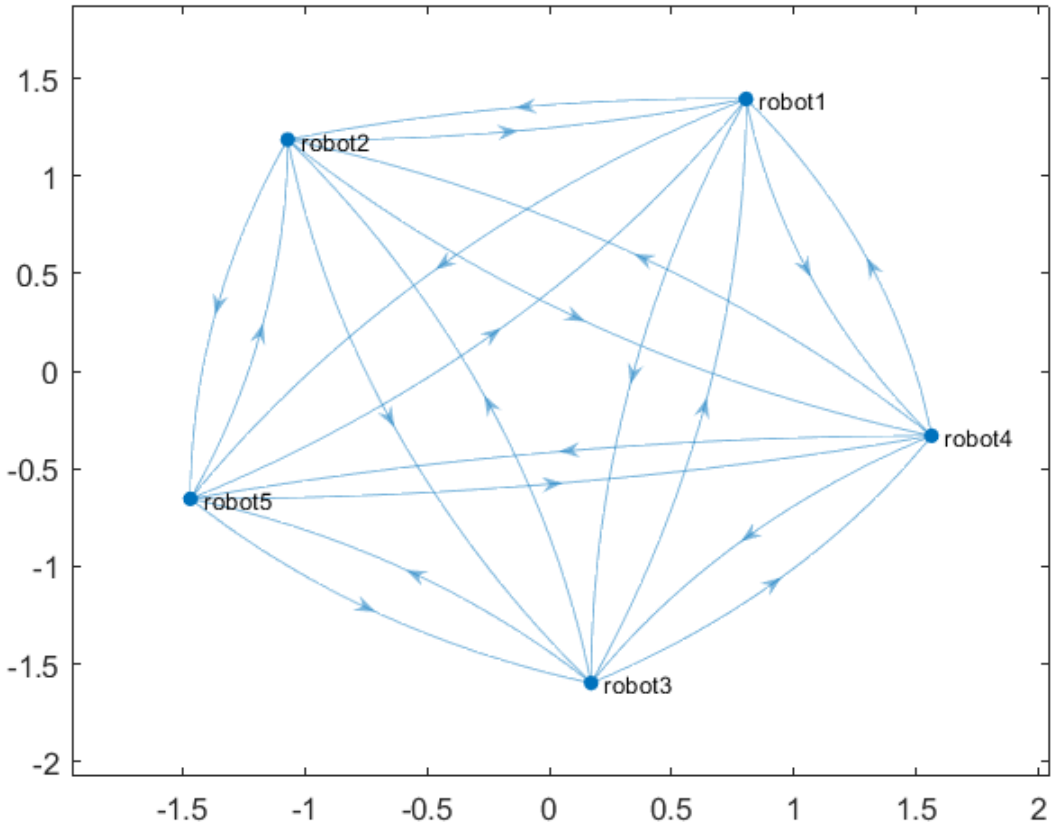

Figure 2: Plot of Graph

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

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

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

3 Consensus-based Control Law:

3.1 Model of a single robot :

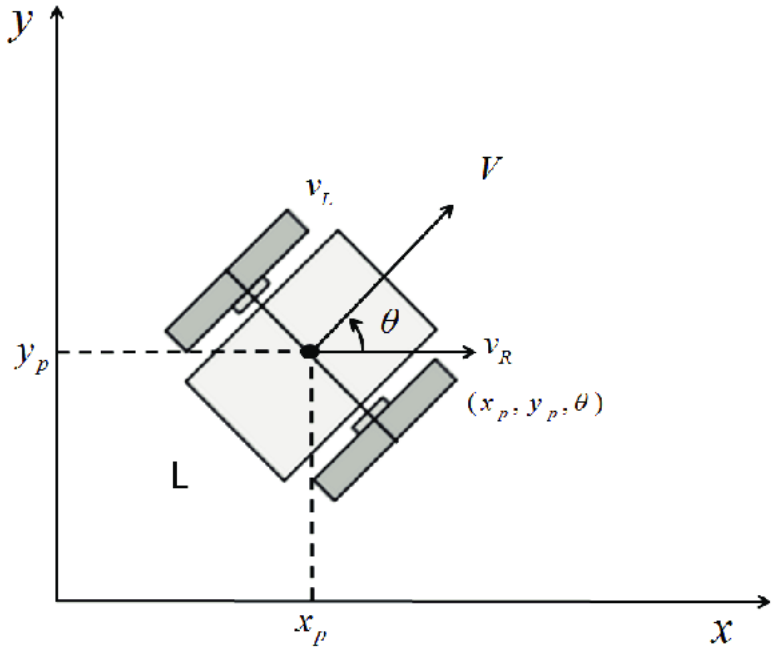

Figure 3: Kinematic Model

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:

$$\begin{bmatrix} \dot{x}_h \\ \dot{y}_h \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -L\sin\theta \\ \sin\theta & L\cos\theta \end{bmatrix}}_{R} \begin{bmatrix} v \\ w \end{bmatrix}$$

Note that: $|R| = L > 0 \implies R$ is invertible $\implies R^{-1}$ exits. Hence we can define new input U such th
$V = R^{-1}U \implies$

$$\begin{bmatrix} \dot{x}_h \\ \dot{y}_h \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{3}$$

3.2 Control Law :

In the cooperative control consensus, robots will share their information about positions or speeds with each other. This information may lead to achieve the common group objective, which in our case of study is to meet in the target location (0,0).

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:

$$\begin{cases} U_{x_i} = -K_{d_i} * (x_{h_i} - x_{f_i}) - \sum_{j=1}^{n} g_{ij} K_{ij} * [(x_{h_i} - x_{h_j}) - (x_{f_i} - x_{f_j})] \\ U_{y_i} = -K_{d_i} * (y_{h_i} - y_{f_i}) - \sum_{j=1}^{n} g_{ij} K_{ij} * [(y_{h_i} - y_{h_j}) - (y_{f_i} - y_{f_j})] \end{cases} \tag{4}$$

Where:
$\mathbf{x}_{h_i}$: The state variable representing the position of robot i at any time t
$\mathbf{x}_{f_i}$: Constant representing the final target tracked by the robot i
$\mathbf{K}_{d_i}$: Gain associated to track final target of robot i
$\mathbf{K}_{ij}$: Gain correlating communication from robot i to robot j.
$\mathbf{g}_{ij}$: Entries of the adjacency matrix

Using the above control inputs, and writing the equations in the state form $\dot{X} = AX$, where X is the state variables vector defined by:

$$X = \begin{bmatrix} x_{h_1} & y_{h_1} & x_{h_2} & y_{h_2} & x_{h_3} & y_{h_3} & x_{h_4} & y_{h_4} & x_{h_5} & y_{h_5} \end{bmatrix}^T \tag{5}$$

$$\begin{bmatrix} \dot{x}_{h_1} \\ \dot{y}_{h_1} \\ \dot{x}_{h_2} \\ \dot{y}_{h_2} \\ \dot{x}_{h_3} \\ \dot{y}_{h_3} \\ \dot{x}_{h_4} \\ \dot{y}_{h_4} \\ \dot{x}_{h_5} \\ \dot{y}_{h_5} \end{bmatrix} = \begin{bmatrix} K_T & 0 & K & 0 & K & 0 & K & 0 & K & 0 \\ 0 & K_T & 0 & K & 0 & K & 0 & K & 0 & K \\ K & 0 & K_T & 0 & K & 0 & K & 0 & K & 0 \\ 0 & K & 0 & K_T & 0 & K & 0 & K & 0 & K \\ K & 0 & K & 0 & K_T & 0 & K & 0 & K & 0 \\ 0 & K & 0 & K & 0 & K_T & 0 & K & 0 & K \\ K & 0 & K & 0 & K & 0 & K_T & 0 & K & 0 \\ 0 & K & 0 & K & 0 & K & 0 & K_T & 0 & K \\ K & 0 & K & 0 & K & 0 & K & 0 & K_T & 0 \\ 0 & K & 0 & K & 0 & K & 0 & K & 0 & K_T \end{bmatrix} \begin{bmatrix} x_{h_1} \\ y_{h_1} \\ x_{h_2} \\ y_{h_2} \\ x_{h_3} \\ y_{h_3} \\ x_{h_4} \\ y_{h_4} \\ x_{h_5} \\ y_{h_5} \end{bmatrix}$$

where $K_T = -4K - K_f$. We used the same gain K between robots for communication, and same target gain K_f, for each robot.
Since $K_T < 0$ for positive K and K_f, the diagonal values of the matrix A of the system are non-positive, so the system is stable.

4 Implementation using MATLAB:

To implement the controlled system proposed above , we have built a function in MATLAB which will receive as an input the initial positions of robots, the gain matrix K_{ij}, the target gain K_f, and the adjacency matrix G. This function will plot the trajectories of the robot's as well as the variation of distance between each robot and the final position as a function of time, which will illustrate the speed of response. The code of the function is presented below:

```
function simulate(initial_pos, gain_matrix, target_gains, adj_m, type)
    % Function that takes control system parameters and plots the response
    % analysis
    %
    % Input:
    %    initial_pos: A 2n-vector of in the form of [x1 y1 x2 y2...] representing
    %                 initial positions of the robots.
    %    gain_matrix: A 2n-by-2n matrix representing the gains of
    %                 communication between each pair of robots. (n being
    %                 the number of robots in the system)
    %    target_gains: A 2n-vector representing gains associated with targets
    %    adj_m: adjacency matrix of the system. Ignored when type
    %           is "full" or "simplified"
    %    type: "full" for fully connected, "simplified for the simplified
    %          system.
    %
    % Output:
    %    no output. Plots the trajectoried of the robots, and the distances
    %    from origin vs time.
```

```
20      %
21      %
22
23      n_robots = floor(size(initial_pos,1)/2);
24      if type == "full"
25          adj_matrix = ones(n_robots)-eye(n_robots);
26
27      elseif type == "simplified"
28          adj_matrix =[
29                      zeros(1,n_robots-1),0;
30                      eye(n_robots-1),zeros(n_robots-1,1)
31                      ];
32
33      else
34              adj_matrix = adj_m;
35      end
36
37
38
39  %     initial_pos = [0 8 -2 -4 3 -1 6 4 -4 7]';
40  %    gain_matrix = 2*ones(n_robots);
41
42      A = zeros(2*n_robots);
43
44      for i=1:2:2*n_robots
45          s1 = 0;
46          s2 = 0;
47          for j=1:2:2*n_robots
48              A(i,j) = A(i,j) + gain_matrix(floor((i+1)/2),floor((j+1)/2))*...
49                  adj_matrix(floor((i+1)/2),floor((j+1)/2));
50              A(i+1,j+1) = A(i+1,j+1) + gain_matrix(floor(i/2+1),floor(j/2+1))*...
51                  adj_matrix(floor(i/2+1),floor(j/2+1));
52
53              s1 = s1+gain_matrix(floor((i+1)/2),floor((j+1)/2))*...
54                  adj_matrix(floor((i+1)/2),floor((j+1)/2));
55              s2 = s2+gain_matrix(floor(i/2+1),floor(j/2+1))*...
56                  adj_matrix(floor(i/2+1),floor(j/2+1));
57          end
58          A(i,i) = A(i,i) - s1 - target_gains(i);
59          A(i+1,i+1) = A(i+1,i+1) - s2 - target_gains(i+1);
60
61      end
62
63      tt = 0:0.001:5;
64      [y,t,x] = initial(ss(A,zeros(2*n_robots,2)...
65          ,eye(2*n_robots),zeros(2*n_robots,2)),initial_pos,tt);
66
67      figure(1)
68      hold on
69      for i = 1:n_robots
70          plot(t,sqrt(x(:,i).^2+x(:,i+1).^2))
71      end
72      legend("Robot_1", "Robot_2", "Robot_3", "Robot_4", "Robot_5")
73      xlabel("Time (s)")
74      ylabel("Distance to Origin")
75      title(sprintf("Distances from Robots to Origin vs Time for kf = %d, k = %.2f",...
```

```
76          target_gains(1),gain_matrix(1,1)))
77
78
79      figure(2)
80      hold on
81      for j=1:2:2*n_robots
82          scatter(x(:,j),x(:,j+1),'.');
83      end
84      legend("Robot_1", "Robot_2", "Robot_3", "Robot_4", "Robot_5")
85      xlabel("x-axis")
86      ylabel("y-axis")
87      title(sprintf("Trajectories of the Robots for kf = %d, k = %.2f",...
88          target_gains(1),gain_matrix(1,1)))
89  end
```

This function will be called twice, once for the fully-connected system, and the other for the simplified system that will be presented in section 5.
Briefly, the function will build up the dynamic matrix of the system (A). It will use the built in functions **ss** and **initial** to simulate the system, then plot the trajectories of the robots and the distance vs. time graph, and calculate the eigenvalues of the dynamic matrix A.

4.1 Results Analysis :

This function presented previously is called multiple times on a fully-connected system, with different gain values. Some of them are presented below:

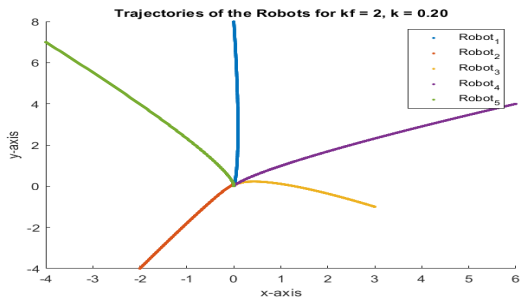

Figure 4: Trajectories for kf=2 and k=0.2

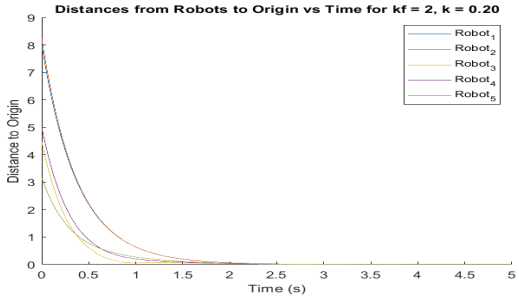

Figure 5: Distances from Robots to Origin vs. Time for kf=2, k=0.2

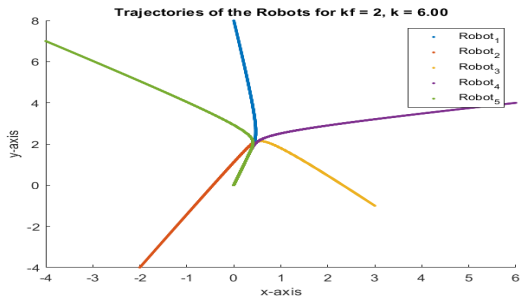

Figure 6: Trajectories for kf=2 and k=6

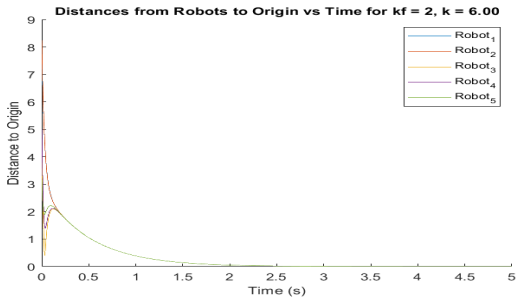

Figure 7: Distances from Robots to Origin vs. Time for kf=2, k=6

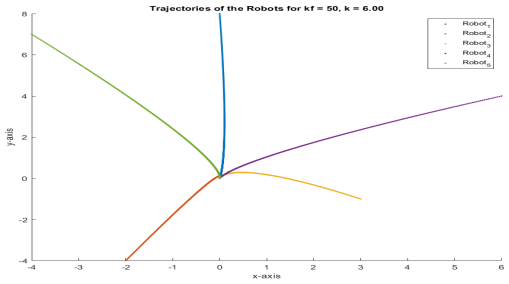

Figure 8: Trajectories for kf=5 and k=6

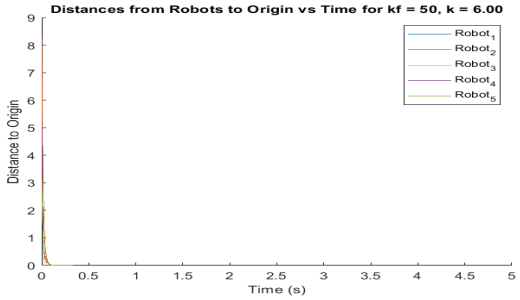

Figure 9: Distances from Robots to Origin vs. Time for kf=50, k=6

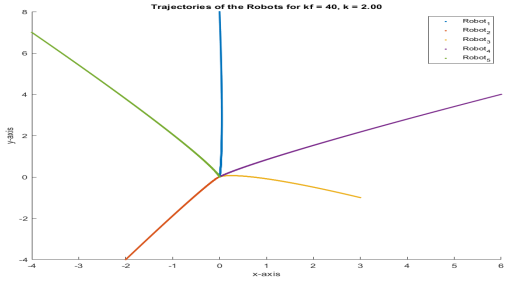

Figure 10: Trajectories for kf=40 and k=2

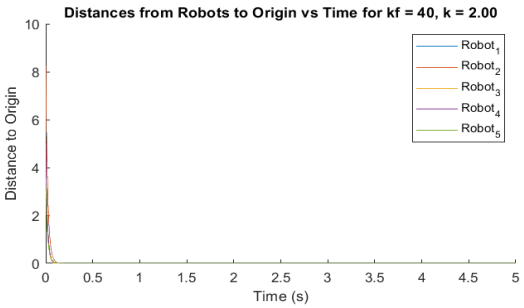

Figure 11: Distances from Robots to Origin vs. Time for kf=40, k=2

As the plots illustrate, bigger gain values result in faster response. However, changing the gains between the robots vs. gains corresponding to reaching the target results in different transient behaviour.
Bigger values of K let the robots give more emphasis to meet instead of reaching the final destination. In the same sense, when we increase the value of K_f the robots give more emphasis to reach the final destination instead.

5 Simplified Graph Proposal:

5.1 Simplified Graph:

To enhance the efficiency of the system while maintaining essential communication information, we reduced the connections between the robots. The following figure represents the new system graph:

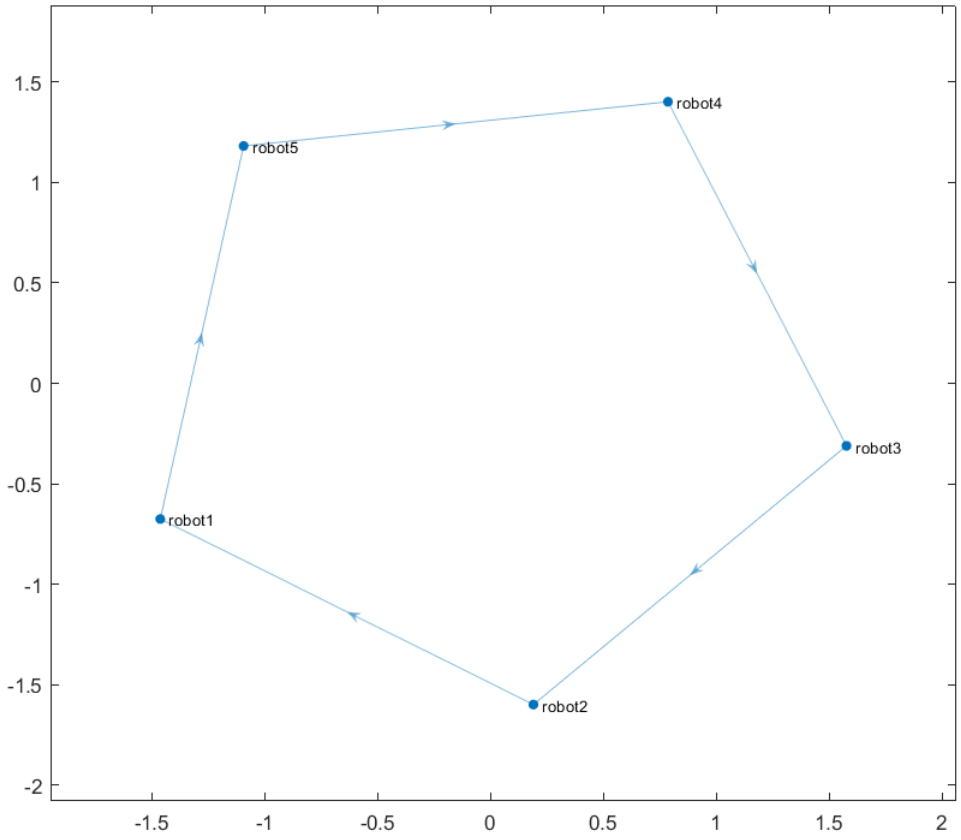

Figure 12: Plot of Simplified Graph

The elements of adjacency (communication) matrix of our system will thus be:

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

5.2 Result Analysis :

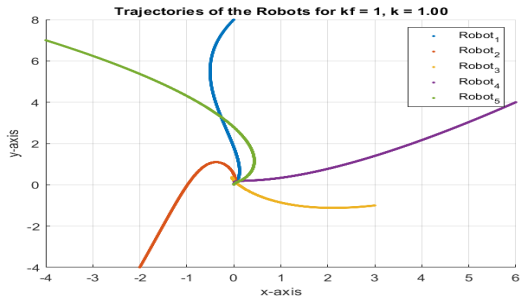

Figure 13: Trajectories for kf=1 and k=1

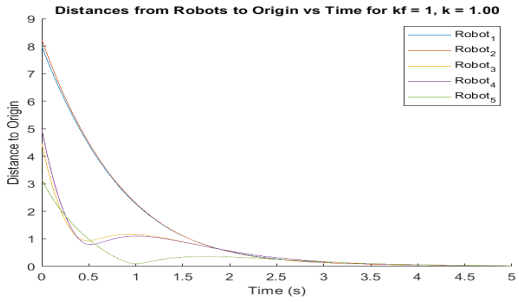

Figure 14: Distances from Robots to Origin vs. Time for kf=1, k=1

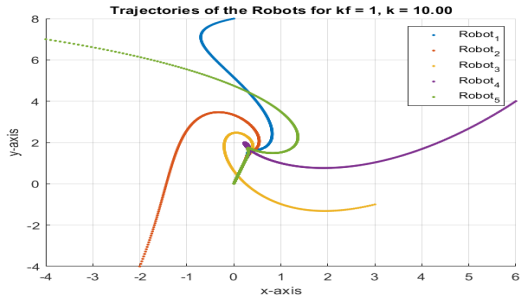

Figure 15: Trajectories for kf=1 and k=10

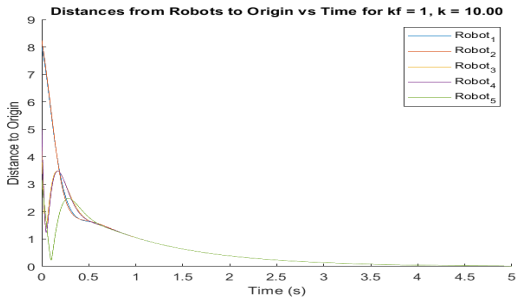

Figure 16: Distances from Robots to Origin vs. Time for kf=1, k=10

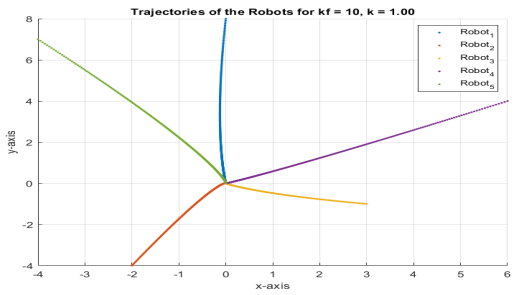

Figure 17: Trajectories for kf=10 and k=1

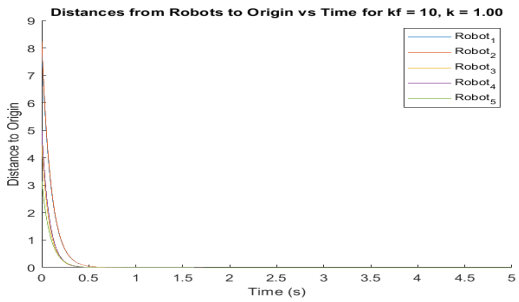

Figure 18: Distances from Robots to Origin vs. Time for kf=50, k=6

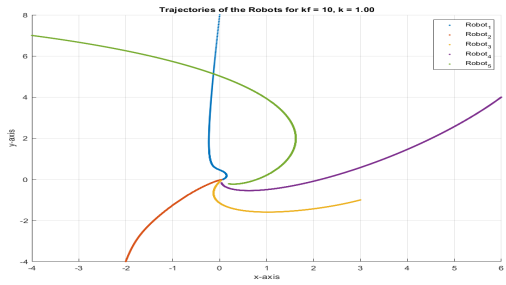

Figure 19: Trajectories for kf=10 and k=1

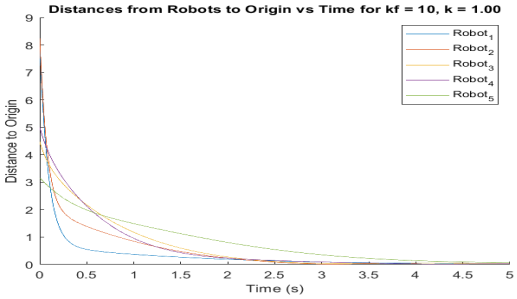

Figure 20: Distances from Robots to Origin vs. Time for kf=10, k=1

Note that the system is simulated for same gains in figures 17 to 20. However, figures 17-18 show the characteristics of the system where all robots have a non-zero K_f, which makes their motion account for reaching the target as well as approaching each others. Figures 19-20 show the characteristics of the system with only one robot having a non-zero gain K_f (robot 1), while the others know nothing about the final target; they are just following each other.

6 Convergence Proof:

The state space representation of our system is :

$$\dot{X} = AX$$

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.

7 Conclusion:

In this mini-project, we have worked on solving the Rendez-vous Problem for 5 robots using a collaborative controller. We have implemented our controller using MATLAB and simulated the performance for different gain values. We also proved the stability and the convergence of our controller , and finally proposed a simplified multi-agent system that achieves the same goal.

References

[1] Lecture Notes, by Dr. Reine Talj-kfoury