Semi-Decentralized Optimal Control of a Cooperative Team of Agents

Navid Rezazadeh
Mechanical Engineering Department
Missouri University of Science and Technology
Rolla, Missouri 65401
Email: nr9q7@mst.edu

***Abstract*—In this project a decentralized optimal control law is proposed for a team of dynamic agents which is also called a multi agent system. The purpose of this controller for the group of agents is to accomplish consensus in a leaderless structure. In this order a semi-decentralized optimal control strategy is designed. The optimal control law is based on the local cost function of each of the agents and this cost function is minimized based on the Hamilton Jacobi Bellman (HJB) set of equations. The dynamic of the agents is governed based on the interaction between agents which is basically the flow of information about states of agents toward each other. Beside this flow of information there is also a local control input to each agent which also plays a roll in controlling dynamics of each agent. The optimal control law essentially effects the system through these two control inputs.It is worth noting that the consensus algorithm is derived in a formal way that is based on conventional multi agents control systems methodologies. Finally, the simulation results are presented in order to show the effectiveness of the proposed optimal control law.**

I. Introduction

Engineered, distributed multi agent networks have posed a number of challenges in terms of their system theoretic analysis and synthesis. Agents in such networks are required to operate in concert with each other in order to achieve system level objectives, while having access to limited computational resources and local communications and sensing capabilities. A good example of multi agent system is unmanned systems networks which draws great amount of attention in multi agent control community. The reason for this importance is that this kind of systems are becoming pretty prevalent. besides, these networks consists of a large number of agents, such as unmanned aerial vehicles, unmanned ground vehicles and unmanned aerial vehicles, and each agent is equipped with tens of sensors which essentially results in a large scale network of sensors and this makes this kind of dynamic systems challenging to control. Some of the applications of the mentioned network can be addressed as home and building automation, intelligent transportation systems, health monitoring and assisting, space exploration, and commercial applications.

In order to deploy these networks, a control algorithm is necessary for the agents so that they become able to cooperate with each other to accomplish a predefined mission. The cooperation of agents in a network of agents is known as formation, networking agreement, swarming or collective motion in different contexts. The extent of the fields where cooperation of the agents is defined makes the control problems pretty challenging in the context of multi agent systems. for example, one of the problems which arises with formation control in presence of uncertainties in the agents which makes it necessary to use adaptive control methods.

There has been a lot of work done in cooperation of a network of agents. However, there are still unsolved problems in this area. In most of the works that has been done so far, first a controller is suggested and then performance and properties of the controller is analyzed. However, there are few works that start from a list of desired specification and then propose a controller based on the list. In this work, an optimal control law is proposed to govern output consensus of a team of agents over a common value based on a predefined system specification where in the context of optimal control it becomes cost function. Using the cost function, it is possible to specify the weights which determines convergence rate to the common value of consensus and the amount of energy which is going to be used.
A centralized implementation of a network of agents means that all agents have access to the information from all other agents in the network or a central brain has access to the local information of all agents and calculates the control signal of each of agent and sends it back to each them. However the method which is proposed in this work is semi decentralized which means that each agent has access to local information of a limited number of agents and calculated it own control signal based in the limited amount of information. This method of implementation reduces the amount of communications and calculations necessary to happen in each agent.

II. Methodology

A. Problem definition

Information structure and neighboring set:
The set of all agents is defined as $\{A = a_i, i = 1, ..., N\}$, where N is the number of total agents. In a network of agents, each agent should work in coordination with other members so it has to know states of other agents. In order to know states of other agents there should exist communication links between agents. However, since the implementation is semi

decentralized, each agent is not connected to all other agents. For each agent there are some specific other agents which it can communicate with. In this work it is assumed that the communication links stay the same all the time, so the topology of network does not change over time. This topology can be defined as set for each agent where it specifies the other agents who has established a connection with the agent. This set is defined as

$$\forall \quad i = 1, ..., N \quad N^i = \left\{ j = 1, ..., N | a_j R a_i \right\} \tag{1}$$

where R designates the existence of a relation between a_i and a_j that the two agents are able to send each other the information about their states.

The mentioned set only provides information about one specific agent but it does not give information about the connections on the whole network. In other to have information of the connections of the agents over the network a matrix is defined as Laplacian of the network. Using this matrix it becomes possible to comprehend information about network structure such as connectedness of network or the neighboring set N^i. The Laplacian matrix is defined as

$$L = [l_{ij}]_{N \times N}; \quad l_{ij} = \begin{cases} O^i & i = j \\ -1 & a_i R a_j \\ 0 & otherwise \end{cases} \tag{2}$$

in which O^i is the number of the links connected to the vertex i and is called the degree of vertex i. This matrix shows the structure of information links existing between the agents. If the network describing this information structure is connected according to the following definition, then the total team cooperation may be expressed in terms of the cooperation among the agents in each cluster.

Connected Graphs: A graph consisting of a vertex set and an edge set is connected if there is a path between any two vertices of it and the path should be in the edge set.

Ring Topology: In this topology each agent is connected to its two neighbors

$$\begin{aligned} \forall \quad i = 2, ..., N \quad N^i = \{i-1, i+1\} \\ N^N = \{N-1, 1\}, \quad N^1 = \{2, N-1\} \end{aligned} \tag{3}$$

1) Model of interaction among the team members: Each of the agents has its own dynamics which is considered to be linear. Each agents follows equation of motion given by

$$\dot{X}^i = A^i X^i + B^i u^i, \quad u^i \in R^m \tag{4}$$

$$Y^i = c^i X^i, \quad Y^i \in R^n \tag{5}$$

in which q, m and n are the dimensions of the state space representing the dynamics of each agent, its control input and output vectors, respectively. The system defined above shows the dynamic of the each agent and as it can be seen the dynamics is isolated for each agent. However, as it was mentioned earlier the team of agents cooperate with each other in order to accomplish a mission. Therefore, there should be some interaction between the agents through their dynamics. This interaction can be considered for each agent through the input channel and based in its available information, which originates form the output of other agents in its neighboring set. This may be described by the following dynamic representation

$$\dot{X}^i = A^i X^i + B^i u^i + B^i \textstyle\sum_{j \in N^i} F^{ij} Y^j \tag{6}$$

$$Y^i = c^i X^i \tag{7}$$

in which $B^i \sum_{j \in N^i} F^{ij} Y^j$ term in the interaction term and F^{ij} is the interaction coefficient. What the interaction terms does is that it injects a control signal to the local dynamics of each agent based on the difference between out put of the agent and its neighbors' outputs.

The above model is an acceptable modelling of interaction between the agents present in the network which shows the effect of flow of information between agents on their dynamics. Based on the proposed modeling the control input to each agent can be written as

$$\hat{u}^i(X^i, Y^j) = u^i(X^i) + \check{u}^i(Y^j) \tag{8}$$

As the above equation suggests the total control input can be divided into two sub control signals. The first one in local to the agents and the second one is totally dependent on the flow of information from the neighboring agents. Separating these two terms helps to identify the interaction more clearly and provides a better insight in the design of a control law which represents the existing interaction among the agents. For simplicity, we can consider the term to be a linear function of neighboring agents, that is, $\check{u}^i(Y^j) = \sum_{j \in N^i} F^{ij} Y^j$.

The above formulation clarifies the fact that the control structure is called semi decentralized. Since $\hat{u}$ is only a function of X^i which is decentralized design, and $\check{u}$ brings the the effect of neighboring agents into the account which is a not-decentralized design. It is not possible to have a design which is fully decentralized since in a context where cooperation of agents is required, the flow of information from agents to each other is inevitable. However the control algorithm is not also fully centralized since flow of of information happens only between the neighboring agents and not between all agents.

2) Definition of a cost function: The main goal of this work is to design an optimal controller. Till now we have defined the dynamic of the agents. The next step is to define a cost function which is going to be used in design of an optimal controller. The general form of the cost function which is going to be used for each agents is

$$\begin{aligned} d^i = \int_0^T [\sum_{j \in N^i} [(Y^i - Y^j)^T Q^{ij} (Y^i - Y^j)] + (u^i)^T R^i u^i] dt \\ + (Y^i)^T E^i Y^i(T) + (F^i)^T Y^i(T) + m^i \end{aligned} \tag{9}$$

in which Q^{ij}, E^i and R^i are symmetric and positive definite matrices, F^i is a vector with proper dimensions, and m^i is a scalar. By minimizing the above cost function one may guarantee that all the agents in a neighboring seat would have the same output vector in steady state. The above cost functions has the term $\sum_{j\in N^i}[(Y^i - Y^j)^T Q^{ij}(Y^i - Y^j)$ which implies that it is desired that one specific agent ant its neighboring agents acquire the same output or in the sense of optimal control the output of these agents become as close as desired based on the weighting function Q^{ij}. In the first glance this term implies that only the agents who are in a neighboring reach the same output at steady state. However due to connectivity of of the information graph any neighboring cluster has at least one common member with one of the other neighboring clusters. This fact ensures that two neighboring sets with a common agent in them are forced to have the same final value at steady state since each of those neighboring sets will have the same final value as the common member. To be more specific assume that member i has a final output vector of Y^{i*}. Any other member k is either inside or outside its neighboring set. In the first scenario, $Y^{k*} \rightarrow Y^{i*}$, in which Y^{k*} is the final output of the k'th member due to the cost minimization. In the second case, due to connectivity of the network there are always a path between every two agents in the network. Therefore, the is path between k'th agent and i'th agents as $\{a_{j_1}, ..., a_{j_m}\}$ where a_{j_1} is neighbor of a_k and a_{j_l} is neighbor of $a_{j_{l-1}}$ and a_{j_m} is a neighbor of a_i. Therefore it can be concluded that $Y^{k*} \rightarrow Y^{j_1*} \rightarrow ... \rightarrow Y^{j_m*} \rightarrow Y^{i*}$. The cost function introduced in 9 is a quadratic function, however as we would know due to dependency of each individual cost function on another agents output and partial availability of information for individual agents, this quadratic optimal control problem cannot be solved using conventional methods such as LQR. This is due to the fact these methods need full access to the information of the group for each agent which is not available in our considered case. Hence to solve this minimization problem we should use a more general solving method which is going to be Hamiltonian Jacobi Bellman equations.

3) Hamiltonian Jacobi Bellman (HJB) eqautions: HJB equations can be used to solve a minimization problem in general form. Assume a system has the dynamic with general form of

$$\dot{X}^i = f^i(t, X^i, u^i) \tag{10}$$

and the cost function to be minimized has the general form of

$$J^i = \int_0^T g^i(t, X^i, u^i)dt + h^i(X^i(T)) \tag{11}$$

If V^i is a value function to be chosen, then the solution to the problem in 10 and 11 is found using the following HJB equations:

$$\begin{aligned} -\frac{\partial V^i}{\partial t} &= \min_{u^i} \Lambda^i(t, X^i, u^i) \\ V^i(T, X^i) &= h^i(X^i(T)) \end{aligned} \tag{12}$$

$$u^{i*}(t, X^i) = \arg\min_{u^i} \Lambda^i(t, X^i, u^i) \tag{13}$$

in which

$$\Lambda^i(t, X^i, u^i) = g^i(t, X^i, u^i) + \frac{\partial V^i}{\partial X^i}(t, X^i) f^i(t, X^i, u^i) \tag{14}$$

To solve the above equation one should solve the PDE equation and find a value function V^i.

4) HJB equation in multi agent optimal control problem: To solve the HJB equations and consequently optimal control problem in case of multi agent systems, first it is necessary to define the dynamics of the agents in a network specifically. In the literature about multi agents systems usually the agents are supposed to have a very simple dynamics where the agents are considered to be masses. All the physical laws about massed are valid in their cases. It means that control signal is inserted to the agents and it causes acceleration. By integrating acceleration it velocity of each agent can be calculated and then by integration the velocity it is possible to specify the location of each agent. Therefore the dynamics if agents in a network can be defined as

$$\begin{cases} \dot{r}^i = v^i \\ \dot{v}^i = u^i + \sum_{j\in N^i} F^{ij} v^j \\ Y^i = v^i \end{cases} \tag{15}$$

in which $r^i \in R^2$ which is the displacement vector of an agent in x and y directions and $v^i \in R^2$ which is the velocity vector of the agent in x and y directions. F^{ij} is consequently a 2×2 matrix. It also can be comprehended that each agent's acceleration is equal to the force which is inserted into the agent locally plus the force that is inserted to the agents based on the its interaction with its neighboring agents. The output of each agent is considered to be its velocity.

For the scenario under consideration given by 15 and 9, the HJB equation will reduce to the following equations

$$\begin{aligned} -\frac{\partial V^i}{\partial t} &= \min_{u^i} \Lambda^i(t, X^i, u^i) \\ V^i(T, X^i) &= v^i(T)^T E^i v^i(T) + \gamma^i(T) \end{aligned} \tag{16}$$

$$\begin{aligned} \Lambda^i(t, X^i, u^i) = &\sum_{j\in N^i} (v^i - v^j)^T Q^{ij}(v^i - v^j) + (u^i)^T R^i u^i \\ &+ \frac{\partial V^i}{\partial v^i}(t, X^i)(u^i + \sum_{j\in N^i} F^{ij} v^j) + \frac{\partial V^i}{\partial r^i}(t, X^i) v^i \end{aligned} \tag{17}$$

The equations above for each agent not only the not only depends on the output of the agent, v^i, but also on the

neighboring agents' outputs, v^j. In this case only the dynamics of each individual agent is considered in its HJB equation and the dynamics of the neighboring agents are assumed as time varying functions. The effects of the these terms related to the output of other agents are canceled through design of interaction terms F^{ij} in the dynamical equations. In other words, the minimization of J^i is accomplished with the assumption that the other agents' dynamics are fixed and the interaction term modifies the effect of v^j in the i'th cost function. So it is possible to minimize d^i only with respect to u^i and

$$u^{i*}(t, X^i) = \arg\min_{u^i} \Lambda^i(t, X^i, u^i) \tag{18}$$

Existence of solution and solution: The existence of solution in the minimization problem depends on controllability and observability of the optimization problem. Based on the equations given in 6 and 7 and 15 we can define

$$\bar{A}^i = \begin{bmatrix} 0_{2\times 2} & I_{2\times 2} \\ 0_{2\times 2} & 0_{2\times 2} \end{bmatrix}, \bar{B}^i = \begin{bmatrix} 0_{2\times 2} \\ I_{2\times 2} \end{bmatrix}, \bar{c}^i = (\bar{B}^i)^T \tag{19}$$

on the other hand r^i does not have any effect on the cost function so we can define

$$A^i = 0_{2\times 2}, B = I_{2\times 2}, B = I_{2\times 2} \tag{20}$$

So we can verify the controllability and observability of the system on therefore the existence of a solution is guaranteed.

In the above minimization problem, the optimal control law is of the form

$$u^{i*} = -\frac{1}{2}(R^i)^{-1}(\frac{\partial V^i}{\partial v^i})^T \tag{21}$$

and the choice for V^i can be specified as

$$V^i = \frac{1}{2}(v^i)^T K^i(t) v^i + \gamma^i(t) \tag{22}$$

in which K^i and γ^i are time varying parameters to be defined.

Lemma 1 *Optimal Control and Finite Horizon* Assume a group of agents whose dynamics are governed by double integrator equations 15 and that operate in the presence of interactions among the based on the neighboring sets as in 15. The interaction coupling terms and the decentralized optimal control law proposed below would minimize the cost function 9, where

$$F^{ij} = 2K^i(t)^{-1}Q^{ij} \tag{23}$$

$$u^{i*} = -\frac{1}{2}(R^i)^{-1}K^i(t)v^i \tag{24}$$

in which K^i satisfies the following Riccati equation

$$-\dot{K}^i = 2N^iQ^{ij} - \frac{1}{2}K^i(R^i)^{-1}K^i, \quad K^i(T) = 2E \tag{25}$$

Proof First note that γ in 17 can be rewritten as

$$\begin{aligned} -\Lambda^i(t, X^i, u^{i*}) + \sum_{j\in N^i}(v^i - v^j)^T Q^{ij}(v^i - v^j) + (u^i)^T R^i u^i \\ + \frac{\partial V^i}{\partial v^i}(t, X^i)(u^i + \textstyle\sum_{j\in N^i} F^{ij}v^j) = -\frac{\partial V^i}{\partial r^i}(t, X^i)v^i \end{aligned} \tag{26}$$

now by replacing V^i, F^{ij} and u^{i*} according to 22, 23 and 24 we get

$$\begin{aligned} -(v^i)^T\frac{\dot{K}^i}{2}v^i - \dot{\gamma}^i = \sum_{j\in N^i}(v^i - v^j)^T Q^{ij}(v^i - v^j) \\ -(v^i)^T\frac{\dot{K}^i}{4}(R^i)^{-1}K^iv^i + (v^i)^T K^i \sum_{j\in N^i} 2(K^i)^{-1}Q^{ij}v^j \end{aligned} \tag{27}$$

Now by equating the corresponding terms in $(v^i)^T v^i$ and v^i the Riccati equation in 25 can be obtained.

Lemma 2 *Sub-Optimal Control and Infinite Horizon* For the group of agents described in Lemma 1 under the infinite horizon scenario the combined control law reduces to

$$\hat{u}^i(X^i, Y^j) = u^i(X^i) + \check{u}^i(Y^j) = \Gamma^i(v^i - \frac{\sum_{j\in N^i} v^j}{N^i}) \tag{28}$$

in which

$$\Gamma^i = -\frac{1}{2}(R^i)^{-1}K^i \tag{29}$$

Proof Note that in infinite horizon case $\dot{K}^i = 0$ and so Riccati equation in 25 reduced to an algebraic Riccati equation given by

$$\begin{aligned} 0 = 2N^iQ^{ij} - \frac{1}{2}K^i(R^i)^{-1}K^i, \quad K^i(T) = 2E \\ \rightarrow \\ -2N^iQ^{ij} = \frac{1}{2}K^i(R^i)^{-1}K^i, \quad K^i(T) = 2E \end{aligned} \tag{30}$$

and consequently we can drive the total optimal control as

$$\begin{aligned} \hat{u}^i(X^i, Y^j) = -\frac{1}{2}(R^i)^{-1}K^i(v^i - \frac{\sum_{j\in N^i} v^j}{N^i}) \\ = \Gamma^i(v^i - \frac{\sum_{j\in N^i} v^j}{N^i}) \end{aligned} \tag{31}$$

III. Simulation, Result and Discussion

In this work we have developed two general control laws of optimal control for finite horizon and infinite horizon and the simulation is done for these two cases. The system which is used for simulation purposes is set of four ground vehicles and their dynamics is governed by the equations given by 15 in two directions in the plane. It means that the input to each agent has two part of x direction and y direction which results in acceleration in two direction of x and y then by integration the output of each agent which is velocity in two directions of x and y can be calculated. According to cost function 9 the goal is to make velocity of all agents as close as possible. So it means that each agent has four states which are acceleration and velocity in two directions of x and y. The topology of the network is in a way that agent 2 and agent 3 are in neighboring of agent 1, agent 1 and agent 4 are in neighboring of agent 2, agent 1 and agent 2 are in neighboring of agent 3, and agent 2 and agent 3 are in neighboring of agent 4 so it can be concluded that $N^i = 2$. For the finite horizon simulation Riccati equation is solved backward using euler method. For the infinite horizon part the $care$ command of MATLAB is used to solve Algebraic Riccati Equation. Figure 1 shows the x component of velocity, v^i of all agents and Figure 2 shows their y component for the infinite horizon case. Figure 3 and Figure 4 show the same trajectory but this time for the finite horizon case. Figure 5 and Figure 6 show the paths generated by the vehicles in the $x-y$ plane. As it can be seen in the infinite horizon case the speed of agents converges to constant value and their trajectory in $x-y$ plan becomes parallel consequently. In the finite horizon case as it can be seen the velocity trajectory of the agents converges to each other but it does not stay constant for all time and the location trajectory of the agents become parallel again. The simulation parameters are given as $Q^{ij} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$, $R^i = I_{2\times 2}$, $N^i = 2$, $T = 30$, $E = I_{2\times 2}$, $r_0^1 = [20 \quad 1]$,$v_0^1 = [5 \quad 8]$, $r_0^2 = [2 \quad 12]$,$v_0^2 = [70 \quad -9]$, $r_0^3 = [18 \quad 8]$,$v_0^3 = [30 \quad -2]$, $r_0^4 = [2 \quad 1]$,$v_0^4 = [10 \quad 9]$.

IV. Conclusion

In this project the problem of consensus of agents on a final value of their output was considered. The goal was to design an optimal controller based on a predefined cost function. The output of optimal controller design consists of two parts. The first part is concerned with the local control input of each agent which is not effected by dynamics of other agents in the group. The second part is concerned with the flow of information between neighboring agents and and specifies an interaction term that effects an individual agent's dynamics based on the states of its neighboring agents. The optimal controller in this work was at first developed for the finite horizon case and then it was extended to the infinite horizon case. It is worth mentioning that most of the works that have been done in multi agent control systems, first they start with a proposed controller and then list its properties, but in this work the design of the controller is initiated with list of desired properties of controller which is the predefined cost function which is a prominent advantage of this work.

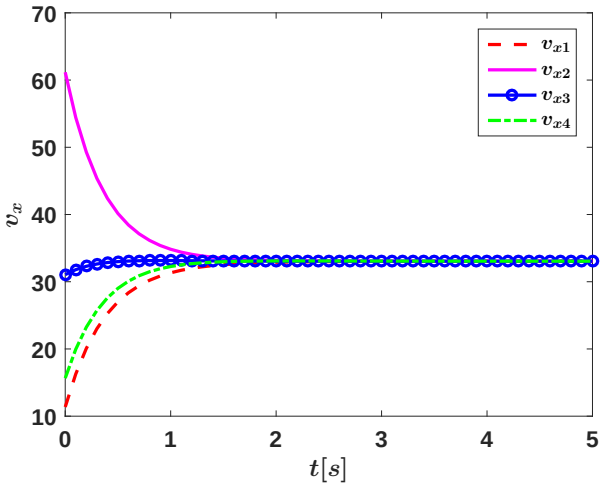

Fig. 1. The x-component of the velocity profile (infinite horizon case)

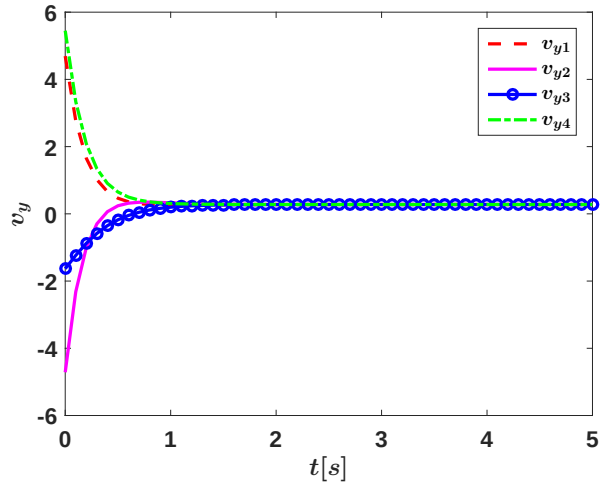

Fig. 2. The y-component of the velocity profile (infinite horizon case)

References

[1] E. S. Kazerooni and K. Khorasani, *"Semi-Decentralized Optimal Control of a Cooperative Team of Agents"*, IEEE International Conference on System of Systems Engineering, April, 2007, pp. 1-7

[2] *"Optimal Control"*, Frank L. Lewis, Draguna L. Vrabie, Vassilis L.Syrmos, Third Edition, 2012, John Wiley & Sons, Inc.

[3] *"Graph theoretic methods in multiagent networks"*, Mesbahi, Mehran and Egerstedt, Magnus, First Edition, 2010, Princeton University Press.

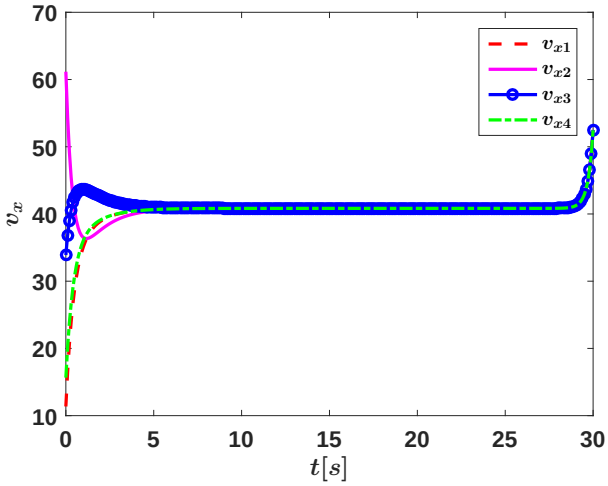

Fig. 3. The x-component of the velocity profile (finite horizon case)

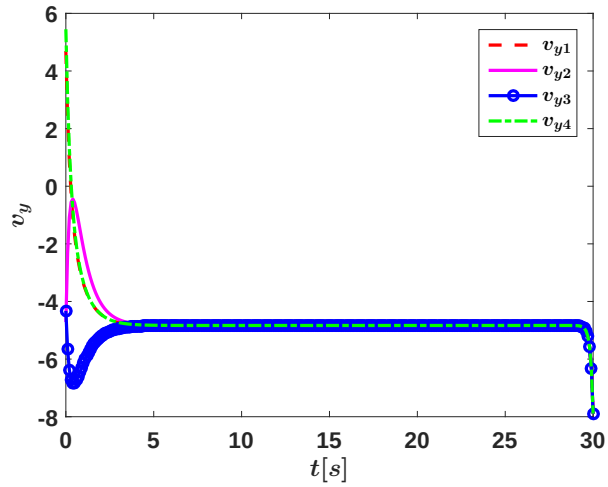

Fig. 4. The y-component of the velocity profile (finite horizon case)

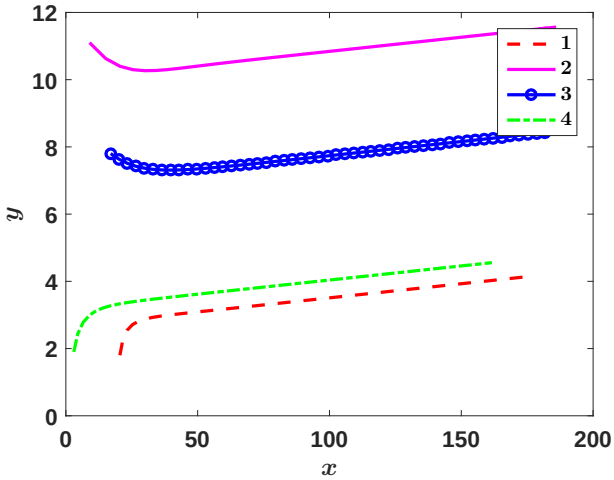

Fig. 5. The $x - y$ path profile (infinite horizon case)

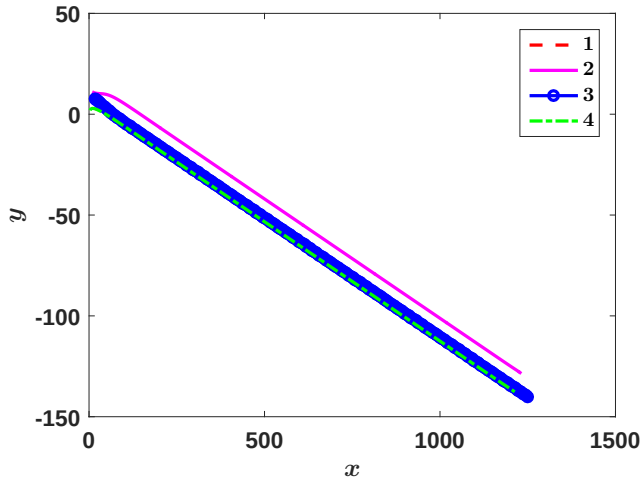

Fig. 6. The $x - y$ path profile (finite horizon case)

Appendix: MATLAB CODE

```
%Infinite horizon simulation

clear; clc

Q=[1 0;0 3]; %Cost function wieghting
R=[1 0;0 1]; %Cost function weighting
N=2; %Number of neighboring agents
E=[1 0;0 1]; %Cost function weighting

v1=[5;8];  %Initial velocity of first agent
v2=[70;-9]; %Initial velocity of second agent
v3=[30;-2]; %Initial velocity of third agent
v4=[10;9]; %Initial velocity of fourth agent

x1=[20;1]; %Initial position of first agent
x2=[2;12]; %Initial position of second agent
x3=[14;8]; %Initial position of third agent
x4=[2;1]; %Initial position of fourth agent


T=5; %Total simulation time
dt=0.1; % time step
t(1)=0;
index=1;

%SteaDY state Optimal control gain calculation
K = care([0 0;0 0],R/sqrt(2),2*N*Q);
lamda=-1/2*R*K;

%Simulation of system
for k=0:dt:T

    %Optimal control gain calculation
    %Acceleration calculation
    v1dot=lamda*(v1-(v2+v3)/2);
    v2dot=lamda*(v2-(v1+v4)/2);
    v3dot=lamda*(v3-(v1+v2)/2);
    v4dot=lamda*(v4-(v2+v3)/2);

    %Positon calculation
    x1=x1+v1*dt;
    x2=x2+v2*dt;
    x3=x3+v3*dt;
    x4=x4+v4*dt;

    %Velocity calculation
    v1=v1+dt*v1dot;
    v2=v2+dt*v2dot;
    v3=v3+dt*v3dot;
    v4=v4+dt*v4dot;

    %data recording

    x1Rec(:,index)=x1;
    x2Rec(:,index)=x2;
    x3Rec(:,index)=x3;
```

```
    x4Rec(:,index)=x4;

    v1Rec(:,index)=v1;
    v2Rec(:,index)=v2;
    v3Rec(:,index)=v3;
    v4Rec(:,index)=v4;

    tRec(index)=k;
    index=index+1;
end

%ploting data

figure; hold on; box on;
p1 = plot(tRec,v1Rec(1,:),'r--');
p2 = plot(tRec,v2Rec(1,:),'m-');
p3 = plot(tRec,v3Rec(1,:),'b-o');
p4 = plot(tRec,v4Rec(1,:),'g-.');
set(p1,'LineWidth',2);
set(p2,'LineWidth',2);
set(p3,'LineWidth',2);
set(p4,'LineWidth',2);
set(gca,'Fontsize',14,'Fontweight','Bold')
xlabel({'\boldmath $t [s]$'},'FontSize',16,'Interpreter','latex');
ylabel({'\boldmath $v_{x}$'},'FontSize',16,'Interpreter','latex');
legend([p1 p2 p3 p4],[{'\boldmath $v_{x1}$'} {'\boldmath $v_{x2}$'} {'\boldmath $v_{x3}$'} {'\bol

figure; hold on; box on;
p1 = plot(tRec,v1Rec(2,:),'r--');
p2 = plot(tRec,v2Rec(2,:),'m-');
p3 = plot(tRec,v3Rec(2,:),'b-o');
p4 = plot(tRec,v4Rec(2,:),'g-.');
set(p1,'LineWidth',2);
set(p2,'LineWidth',2);
set(p3,'LineWidth',2);
set(p4,'LineWidth',2);
set(gca,'Fontsize',14,'Fontweight','Bold')
xlabel({'\boldmath $t [s]$'},'FontSize',16,'Interpreter','latex');
ylabel({'\boldmath $v_{y}$'},'FontSize',16,'Interpreter','latex');
legend([p1 p2 p3 p4],[{'\boldmath $v_{y1}$'} {'\boldmath $v_{y2}$'} {'\boldmath $v_{y3}$'} {'\bol

figure; hold on; box on;
p1 = plot(x1Rec(1,:),x1Rec(2,:),'r--');
p2 = plot(x2Rec(1,:),x2Rec(2,:),'m-');
p3 = plot(x3Rec(1,:),x3Rec(2,:),'b-o');
p4 = plot(x4Rec(1,:),x4Rec(2,:),'g-.');
set(p1,'LineWidth',2);
set(p2,'LineWidth',2);
set(p3,'LineWidth',2);
set(p4,'LineWidth',2);
set(gca,'Fontsize',14,'Fontweight','Bold')
xlabel({'\boldmath $x$'},'FontSize',16,'Interpreter','latex');
ylabel({'\boldmath $y$'},'FontSize',16,'Interpreter','latex');
legend([p1 p2 p3 p4],[{'\boldmath $1$'} {'\boldmath $2$'} {'\boldmath $3$'} {'\boldmath $4$'}],'I


%Finite horizon simulation

clear; clc
```

```
Q=[1 0;0 3]; %Cost function wieghting
R=[1 0;0 1]; %Cost function weighting
N=2; %Number of neighboring agents
E=[1 0;0 1]; %Cost function weighting

v1=[5;8];  %Initial velocity of first agent
v2=[70;-9]; %Initial velocity of second agent
v3=[30;-2]; %Initial velocity of third agent
v4=[10;9]; %Initial velocity of fourth agent

x1=[20;1]; %Initial position of first agent
x2=[2;12]; %Initial position of second agent
x3=[14;8]; %Initial position of third agent
x4=[2;1]; %Initial position of fourth agent


T=30; %Total simulation time
dt=0.1; % time step
t(1)=0;
index=1;

K = 2*E; %initial condition for calculating optimal conrol gain

%Optimal control gain calculation
for k=T:-dt:0

    Kdot=2*N*Q-1/2*K(:,:,index)*R*K(:,:,index);
    K(:,:,index+1)=K(:,:,index)+Kdot*dt;
    tRec(index)=k;
    index=index+1;
end

t(1)=0;
index=1;


%Simulation of system
for k=0:dt:T

    %Interaction term calculation
    F= 2*K(:,:,length(K)-index+1)^-1*Q;

    %Optimal control gain calculation
    u1=-1/2*R^1*K(:,:,length(K)-index+1)*v1;
    u2=-1/2*R^1*K(:,:,length(K)-index+1)*v2;
    u3=-1/2*R^1*K(:,:,length(K)-index+1)*v3;
    u3=-1/2*R^1*K(:,:,length(K)-index+1)*v4;

    %Acceleration calculation
    v1dot=u1+F*(v2+v3);
    v2dot=u2+F*(v1+v4);
    v3dot=u3+F*(v1+v2);
    v4dot=u3+F*(v2+v3);

    %Positon calculation
    x1=x1+v1*dt;
    x2=x2+v2*dt;
    x3=x3+v3*dt;
    x4=x4+v4*dt;
```

```
    %Velocity calculation
    v1=v1+dt*v1dot;
    v2=v2+dt*v2dot;
    v3=v3+dt*v3dot;
    v4=v4+dt*v4dot;

    %data recording
    x1Rec(:,index)=x1;
    x2Rec(:,index)=x2;
    x3Rec(:,index)=x3;
    x4Rec(:,index)=x4;

    v1Rec(:,index)=v1;
    v2Rec(:,index)=v2;
    v3Rec(:,index)=v3;
    v4Rec(:,index)=v4;

    tRec(index)=k;
    index=index+1;
end

%ploting data

figure; hold on; box on;
p1 = plot(tRec,v1Rec(1,:),'r--');
p2 = plot(tRec,v2Rec(1,:),'m-');
p3 = plot(tRec,v3Rec(1,:),'b-o');
p4 = plot(tRec,v4Rec(1,:),'g-.');
set(p1,'LineWidth',2);
set(p2,'LineWidth',2);
set(p3,'LineWidth',2);
set(p4,'LineWidth',2);
set(gca,'Fontsize',14,'Fontweight','Bold')
xlabel({'\boldmath $t [s]$'},'FontSize',16,'Interpreter','latex');
ylabel({'\boldmath $v_{x}$'},'FontSize',16,'Interpreter','latex');
legend([p1 p2 p3 p4],[{'\boldmath $v_{x1}$'} {'\boldmath $v_{x2}$'} {'\boldmath $v_{x3}$'} {'\bol

figure; hold on; box on;
p1 = plot(tRec,v1Rec(2,:),'r--');
p2 = plot(tRec,v2Rec(2,:),'m-');
p3 = plot(tRec,v3Rec(2,:),'b-o');
p4 = plot(tRec,v4Rec(2,:),'g-.');
set(p1,'LineWidth',2);
set(p2,'LineWidth',2);
set(p3,'LineWidth',2);
set(p4,'LineWidth',2);
set(gca,'Fontsize',14,'Fontweight','Bold')
xlabel({'\boldmath $t [s]$'},'FontSize',16,'Interpreter','latex');
ylabel({'\boldmath $v_{y}$'},'FontSize',16,'Interpreter','latex');
legend([p1 p2 p3 p4],[{'\boldmath $v_{y1}$'} {'\boldmath $v_{y2}$'} {'\boldmath $v_{y3}$'} {'\bol

figure; hold on; box on;
p1 = plot(x1Rec(1,:),x1Rec(2,:),'r--');
p2 = plot(x2Rec(1,:),x2Rec(2,:),'m-');
p3 = plot(x3Rec(1,:),x3Rec(2,:),'b-o');
p4 = plot(x4Rec(1,:),x4Rec(2,:),'g-.');
set(p1,'LineWidth',2);
set(p2,'LineWidth',2);
set(p3,'LineWidth',2);
set(p4,'LineWidth',2);
```

```
set(gca,'Fontsize',14,'Fontweight','Bold')
xlabel({'\boldmath $x$'},'FontSize',16,'Interpreter','latex');
ylabel({'\boldmath $y$'},'FontSize',16,'Interpreter','latex');
legend([p1 p2 p3 p4],[{'\boldmath $1$'} {'\boldmath $2$'} {'\boldmath $3$'} {'\boldmath $4$'}],'I
```