OPTIMAL CONSESNUS OF MULTIAGENT SYSTEMS USING REINFORCEMENT LEARNING APPROACH

Summer Research Internship Project Report

Under the guidance of:

Dr Surajit Panja

- Assistant Professor, Dept of ECE,

Indian Institute of Information Technology, Guwahati

Done By:

Aniirudh Ramesh

SASTRA Deemed University

TABLE OF CONTENTS

ABSTRACT: This is a project about using reinforcement learning approach to receive optimal consensus for a multiagent system. It begins with background information on graph theory and mathematical preliminaries utilized in multi-agent systems, and explains consensus and reinforcement learning algorithms. Finally, Simulations are performed using MATLAB and the results are presented.

INTRODUCTION: A Multiagent system is one in which multiple agents are part of the same system and interact with each other to produce a fruitful output. This topic is acknowledged over the past years due to its wide applications in unmanned aerial vehicles (UAVs), robotics in manufacturing and military, space missions using satellites, employees in a company completing specific task, and so forth. Cooperative control consists of several diverse yet related topics such as consensus, flocking and formation algorithms. Consensus is defined to be a group of agents reaching convergence to a mutual set of goals by collaborating with each other thorough sensing or communication network. Consensus problem showed up in many applications of cooperative control such as swarming, where an agreement is reached with the closest surrounding neighbors. Optimal Consensus is the optimization of the cost function that involves the consensus function, subject to constraints like minimum energy, minimum convergence time, etc. Reinforcement learning is a method of machine learning and optimization, which has been used to achieve optimal control and consensus of the system. The goal of this project is to implement a simulation of the same using MATLAB and present the results.

MATHEMATICAL BACKGROUND: This project involves a lot of mathematical concepts including graph theory, matrix theory, Kronecker product and reinforcement learning, which are given below.

1. Graph theory:

Definition: Graph theory a building block in the communication network for the MAS. In this report, the behaviors and connections of the MAS are possible through links of a communication graph; hence, covering graph theory becomes essential. Information between agents is either sent from one agent to another (directional) or sent both ways (bidirectional). Graph theory is a crucial mean to portray the MAS network topology and interaction between the nodes. There are many types of graphs like signed/unsigned, directed/undirected graphs.

2. Graphs and Matrices:

Degree, Adjacency and Laplacian Matrices of Graph:

In an undirected graph G, a degree of a vertex is denoted by $\mathbb{d}(vi)$, and it is equivalent to the number of vertices adjacent to a vertex (vi) in the graph. Degree matrix, denoted by $D(G)$, is a diagonal matrix with diagonal elements equal to the degree of each vertex.

Adjacency matrix is a great attribute to define a communication graph since it consists of information regarding the agents and their connections [8]. The adjacency matrix of an undirected graph is denoted by $A(G)$, and is a symmetric $n \times n$ matrix defined as

$$\boldsymbol{A}(\boldsymbol{G}) = \{1 \quad vivj \in E \\ 0 \quad otherwise\}$$

Laplacian: Graph Laplacian is another graph, which shows important properties of the graph. Graph Laplacian is denoted by $L(G)$, and defined as $L(G) = D(G) - A(G)$.

Where $D(G)$, is the Degree Matrix, and $A(G)$, is the Adjacency Matrix

Kronecker Product: Kronecker product is an operation performed on two matrices or vectors with a random size in which results in a block matrix. This operation is denoted with symbol, $\otimes$. For instance, assume matrix $\boldsymbol{A}$ is $n \times m$ and matrix $\boldsymbol{B}$ is $p \times q$; therefore, the Kronecker product of $\boldsymbol{A} \otimes \boldsymbol{B}$ is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = [A11\boldsymbol{B} \ldots A1m\boldsymbol{B}$$

$$\vdots \quad \ddots \quad \vdots$$

$$An1\boldsymbol{B} \ \ldots \ Anm\boldsymbol{B}] \text{ with a size } np \times mn.$$

PROBLEM DEFINTION: The aim of the project is to perform a MATLAB simulation of the optimal consensus of a Multiagent system achieved using reinforcement learning. For this, we need to define the mathematical equations that govern the working of the system.

Consensus equations:

Consensus is an important topic in area of cooperative control of MAS, which is implemented in both formation control as well as flocking algorithm. Consensus is defined as an agreement between multiple agents on a certain shared variable or a common goal by group interaction via communication links or sensors; when this condition is satisfied, it is stated that agents have reached a consensus.

Consider a system, whose dynamics is as follows:

$$\dot{x}_i = \boldsymbol{A}x_i + \boldsymbol{B}u_i$$

$$y_i = \boldsymbol{C}x_i, i = 1, \ldots, N$$

In this model, the state, control and the output for the ith agent are represented as $xi \in \mathbb{R}n$, $ui \in \mathbb{R}p$, and $yi \in \mathbb{R}q$. In addition, $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are constant matrices and their dimensions change based on the number of agents to satisfy the matrix multiplication.

Consensus is achieved when:

$$u_i(t) = \sum_{j \in N_i} a_{ij}\left[x_j(t) - x_i(t)\right]$$

Which is the same as follows:

Since $\dot{x}(t) = ui(t)$, continuous-time (CT) consensus algorithm in [8] is represented by the linear system as

$$\dot{x}_i(t) = \sum_{j \in N_i} a_{ij}\left[x_j(t) - x_i(t)\right] \quad \text{(local voting protocol)} \tag{2}$$

$i = \{1,2, \dots, n\}$ – number of agents

$A(t) = \left[a_{ij}\right]$ – adjacency matrix for the graph,

$N_i(t) = \{ j \in V : a_{ij} \neq 0 \}$ – set of neighbors for agent i

$x_i(t)$ – position of agent i at time t

$x_j(t)$ – position of agent j at time t

$\dot{x}_i(t)$ – velocity of agent i at time t

Further expansion of Equation (2) and rewriting it including the graph Laplacian matrix result in,

$$\dot{x}_i = \sum_{j \in N_i} a_{ij}\left[x_j(t) - x_i(t)\right] = -x_i \sum_{j \in N_i} a_{ij} + \sum_{j \in N_i} a_{ij} x_j = -d_i x_i + [a_{i1} \dots a_{iN}] \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

d_i is the in-degree of the diagonal matrix, and $[x_1, \dots, x_N]$ is the state vector. Therefore, the system dynamic for CT consensus becomes,

$$\dot{x} = -Dx + Ax = -(D - A)x$$

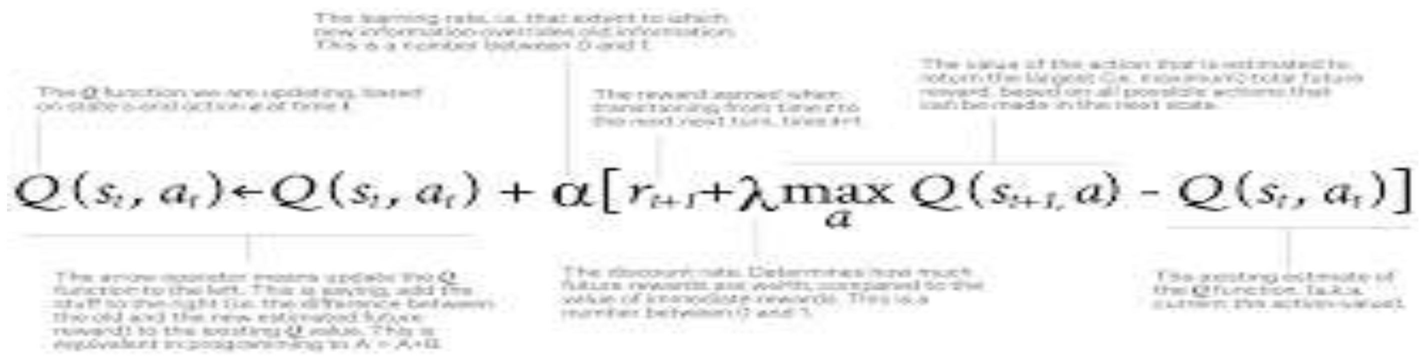
$$\dot{x} = -Lx \;\; \text{Closed loop Dynamics} \tag{3}$$

and the control input is $u = -Lx$

Hence, the control input, which is dictated by the Laplacian matrix is what makes the system achieve consensus. Also the speed of convergence is given by:

r = 1/ e2, where e2 is the 2nd smallest eigen value of the Laplacian Matrix

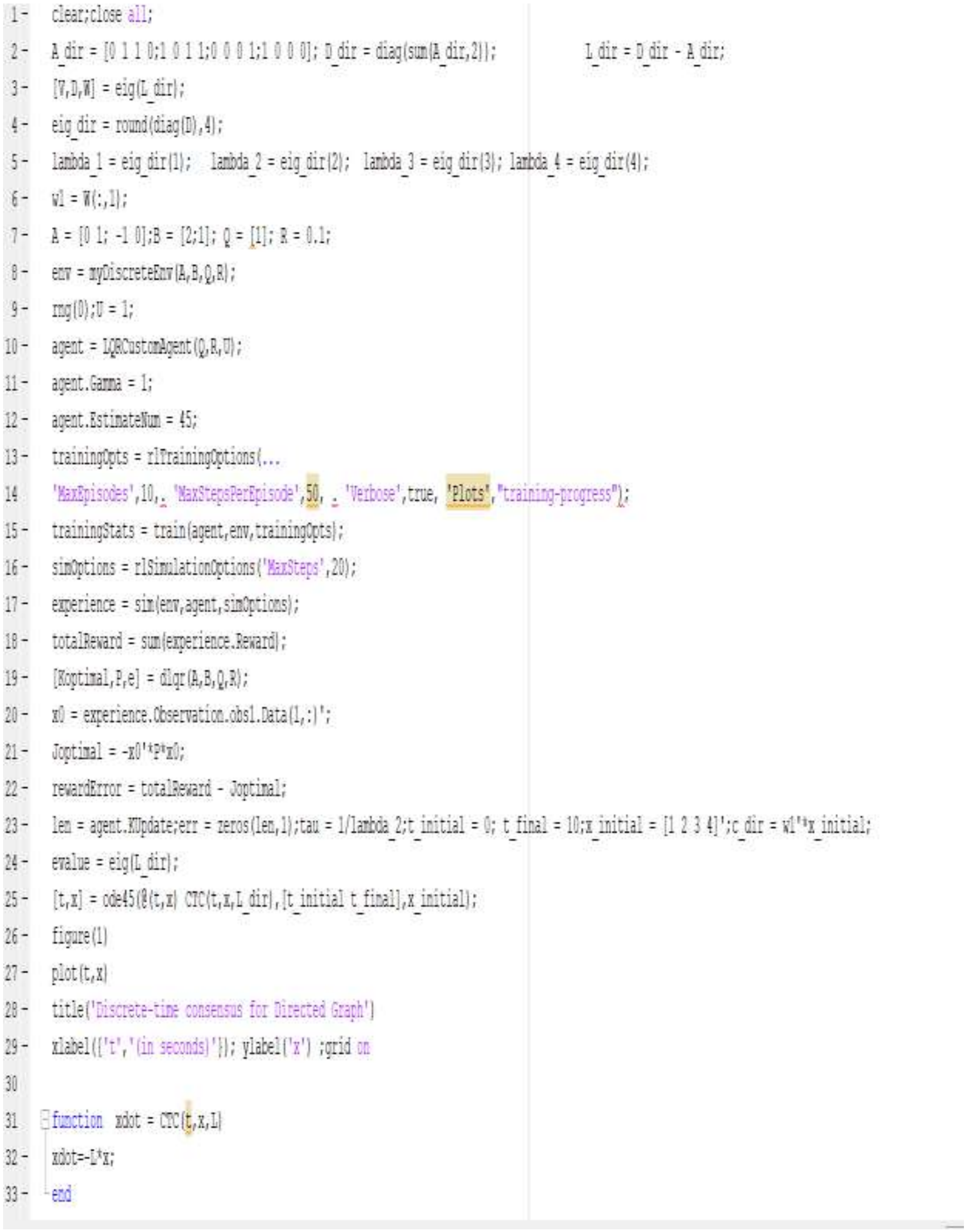
Reinforcement Learning:


The learning rate, i.e. that extent to which new information overrides old information. This is a number between 0 and 1.
The Q function we are updating, based on state s and action a at time t.
The reward earned when transitioning from time t to the next next turn, time t+1.
The value of the action that is estimated to return the largest (i.e. maximum) total future reward, based on all possible actions that can be made in the next state.
The arrow operator means update the Q function to the left. This is saying, add the stuff to the right (i.e. the difference between the old and the new estimated future reward) to the existing Q value. This is equivalent in programming to A = A+B.
The discount rate. Determines how much future rewards are worth, compared to the value of immediate rewards. This is a number between 0 and 1.
The existing estimate of the Q function, (a.k.a. current the action-value).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

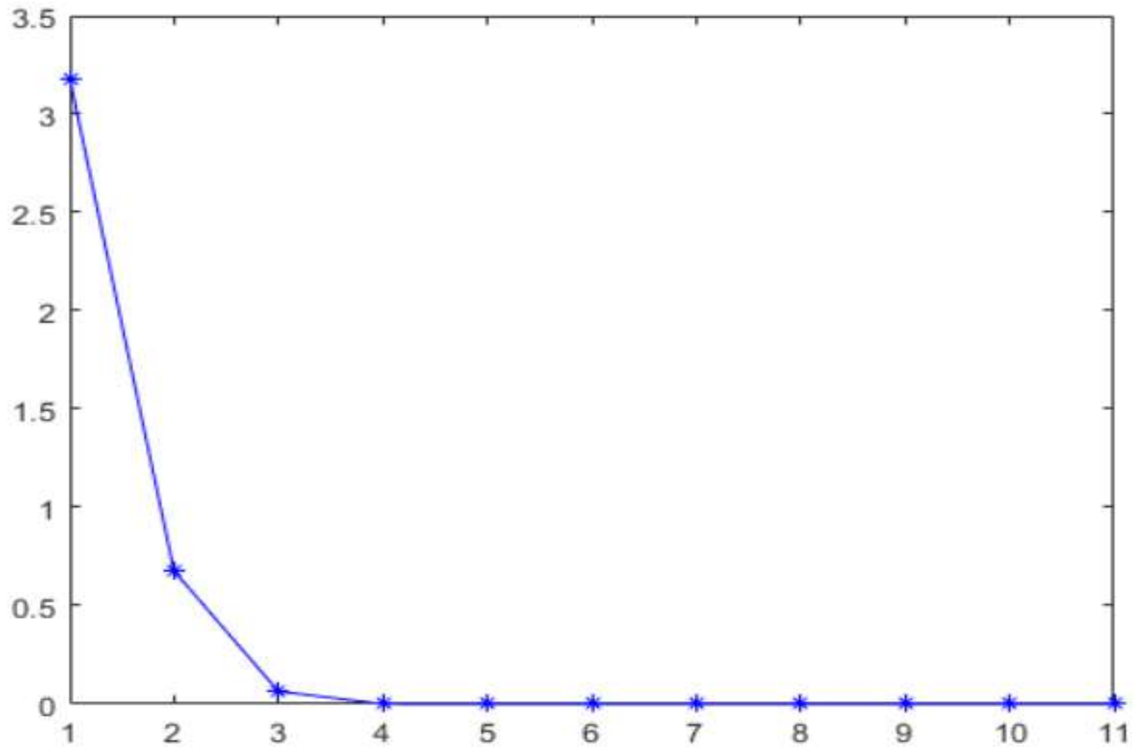
```
Initialize a policy $\pi'$ arbitrarily
Repeat
    $\pi \leftarrow \pi'$
    Compute the values using $\pi$ by
        solving the linear equations
            $V^{\pi}(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^{\pi}(s')$
    Improve the policy at each state
        $\pi'(s) \leftarrow \arg\max_a (E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi}(s'))$
Until $\pi = \pi'$
```

METHODOLOGY/ MATLAB CODE: All simulations were performed using MATLAB and the Reinforcement Learning toolbox was extensively used. The mathematical equations were converted to MATLAB Code and the cost function was minimized using pre-defined MATLAB functions in the Reinforcement Learning Toolbox. The MATLAB function “ode45” is used to solve the linear differential equation that governs the system dynamics. The code for which, is as follows:



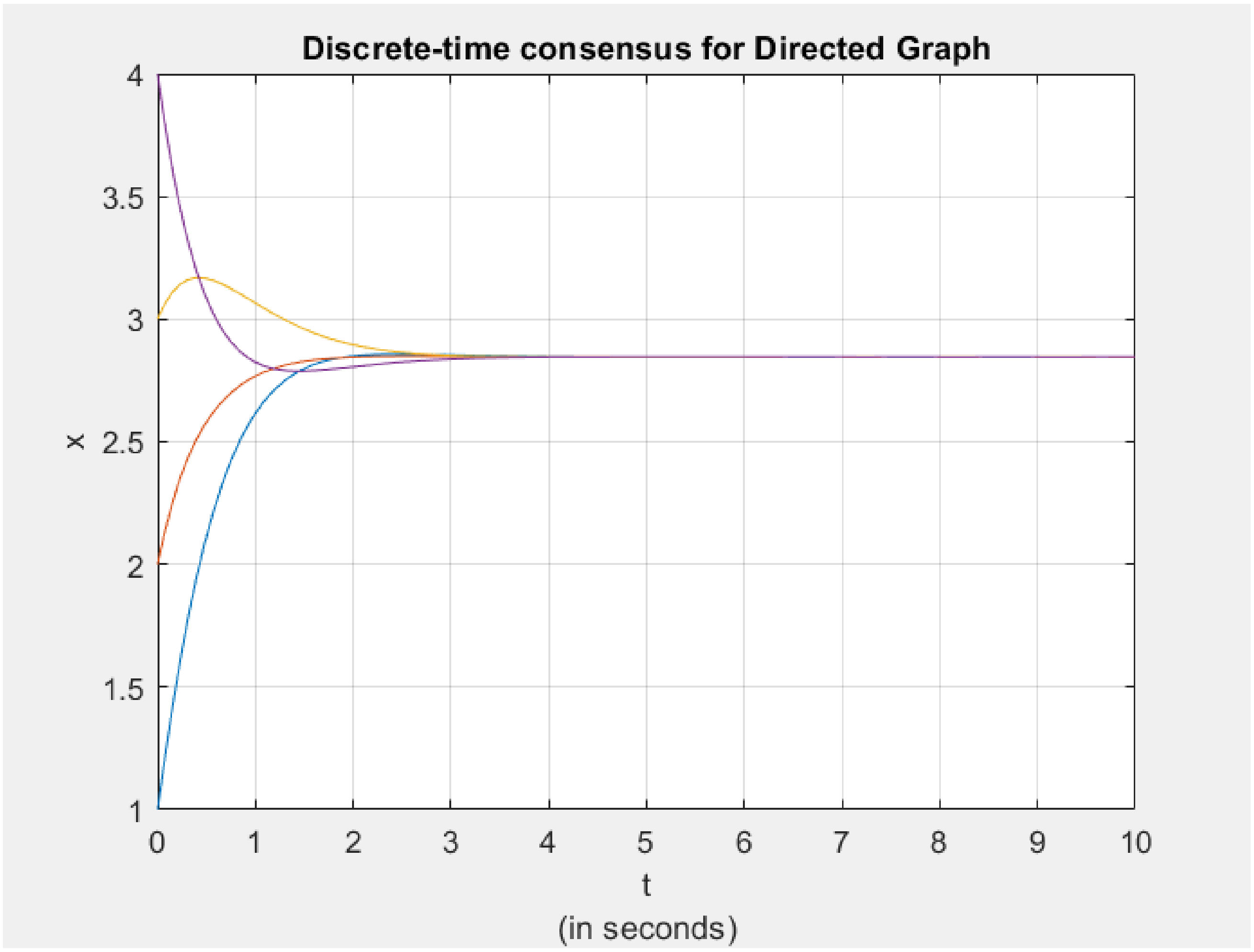
```matlab
clear;close all;
A_dir = [0 1 1 0;1 0 1 1;0 0 0 1;1 0 0 0]; D_dir = diag(sum(A_dir,2));              L_dir = D_dir - A_dir;
[V,D,W] = eig(L_dir);
eig_dir = round(diag(D),4);
lambda_1 = eig_dir(1);   lambda_2 = eig_dir(2);  lambda_3 = eig_dir(3); lambda_4 = eig_dir(4);
w1 = W(:,1);
A = [0 1; -1 0];B = [2;1]; Q = [1]; R = 0.1;
env = myDiscreteEnv(A,B,Q,R);
rng(0);U = 1;
agent = LQRCustomAgent(Q,R,U);
agent.Gamma = 1;
agent.EstimateNum = 45;
trainingOpts = rlTrainingOptions(...
'MaxEpisodes',10,. 'MaxStepsPerEpisode',50, . 'Verbose',true, 'Plots',"training-progress");
trainingStats = train(agent,env,trainingOpts);
simOptions = rlSimulationOptions('MaxSteps',20);
experience = sim(env,agent,simOptions);
totalReward = sum(experience.Reward);
[Koptimal,P,e] = dlqr(A,B,Q,R);
x0 = experience.Observation.obs1.Data(1,:)';
Joptimal = -x0'*P*x0;
rewardError = totalReward - Joptimal;
len = agent.KUpdate;err = zeros(len,1);tau = 1/lambda_2;t_initial = 0; t_final = 10;x_initial = [1 2 3 4]';c_dir = w1'*x_initial;
evalue = eig(L_dir);
[t,x] = ode45(@(t,x) CTC(t,x,L_dir),[t_initial t_final],x_initial);
figure(1)
plot(t,x)
title('Discrete-time consensus for Directed Graph')
xlabel({'t','(in seconds)'}); ylabel('x') ;grid on

function  xdot = CTC(t,x,L)
xdot=-L*x;
end
```

RESULT: The simulation results are given below. Thus, consensus of the MAS is achieved using Reinforcement Learning Approach.



This is the result for the simulation for a single system, whose optimal gain ‘K’ (in case of MAS optimal gain will be the optimal Laplacian) , achieved through reinforcement learning method, is compared with LQR Method, through 10 iterations. The difference is negligible, hence optimal gain has been obtained using RL..

PTO



This represents the state values of 4 agents achieving consensus by finding the optimal Laplacian through RL method, hence achieving optimal consensus.

REFERENCES:

1. "Data-Based Optimal Control of Multiagent Systems: A Reinforcement Learning Design Approach" – Zhang et al , IEEE Cynernetics