A Deep Dive Into Multi-Agent Systems & Applications of Consensus Algorithms

Jason L Kootsher
MSEE University of Colorado, Denver, Colorado, 80204, United States

I. Introduction

Multi agent control systems is a ground breaking field of study that is beginning to be utilized heavily in the modern age. From GPS auto navigation to autonomous vehicles, the notion of "follow the leader", and consensus decision making, is at the core of these technologies. This report will focus on two algorithms at the heart of these control algorithms, and then give an analysis of two papers as well as duplicating their results.

II. Normalized Gradient & Recursive Least Squares

A. Summary

The normalized gradient and recursive least squares algorithms are means by which the minimum of a functional can be calculated. This is important in control theory as the means by which a stable controller is developed require that mathematically, the functional under control be brought to a minimum. Ideally, this would be a global minimum, but local minimums will achieve stability as well.

The major difference between the normalized gradient algorithm (NGA) and recursive least squares (RLS) is in how each operate on their respective functional spaces, and how each is computed numerically. Otherwise both are, from a procedural perspective, identical.

NGA acts as trace along functional curves, with each iteration having no memory of the previous iterations, and is setup to point in the direction of the functional minimum. The fact that this algorithm requires no previous knowledge of iterations is both a positive and a negative. Positively, it reduces calculation time and gives very good insight to the functional behavior. The downside to this though is that because there is no memory, it is impossible to determine if the controller has settled on a local, or the global minimum. Due to this, parameter convergence, while existent, may not approach the values established in the control law. From a multi agent controls perspective, these gain differences can result in undesired behavior, such as pushing each controller toward an undesired position or velocity.

RLS, while very similar to NGA, has enough differences to where it is far more powerful of an algorithm. The RLS algorithm does have memory, and operates not on the functional directly, but on the space that the functional resides within. Instead of tracing curves of the functional, the space local to the functional is rotated within the global space under an invariant transformation (eigen) such that the projection of the functional space on the global space results in a collapse to the null space. Since the space is minimized globally, RLS gives a global minimum. The biggest issue with this algorithm is that computationally it requires significantly more processing, especially for more complex controllers, and it is possible to fail computationally altogether if the space in question is singular. For singular matrices, convergence to a local minimum, and therefore overall convergence is still possible, but not with the RLS algorithm.

B. Analysis

The following analysis was performed on "Part 1" of Appendix A, the "System Identification Problem" (SIP).

A sample size for the plots for SIP are listed below. To generate the full plots, please utilize the code in Appendix B.

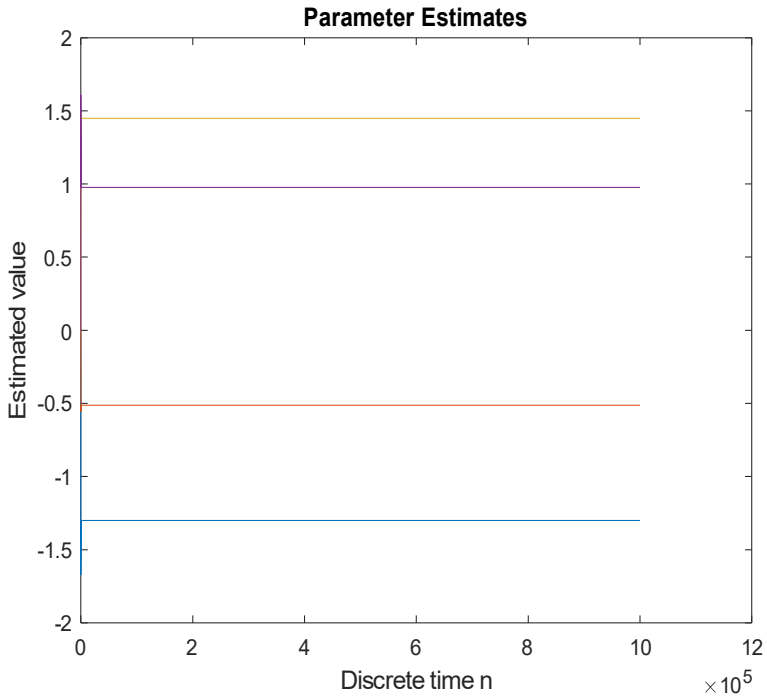

Figure 1A: NGA Parameter Estimates Over 1E6 Samples

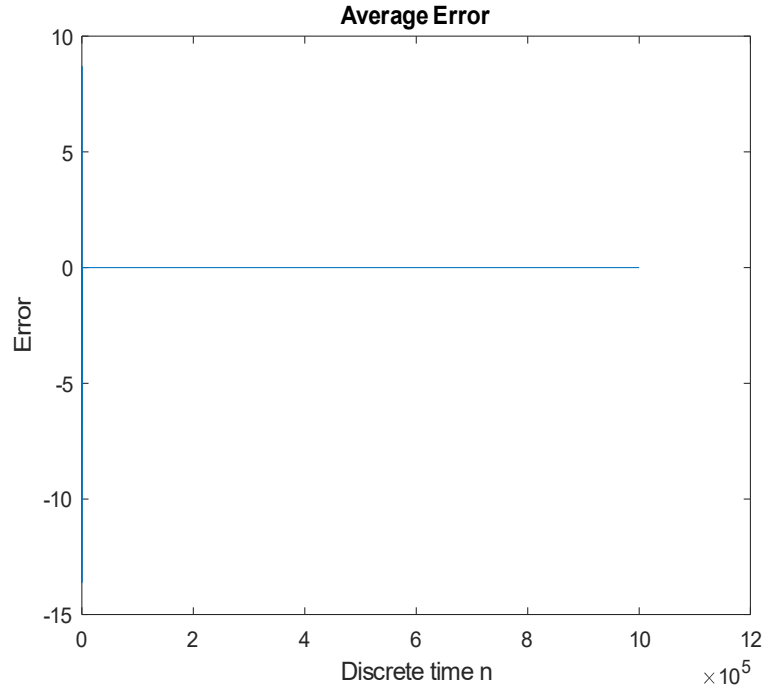

Figure 1B: NGA Error in Parameter Estimates

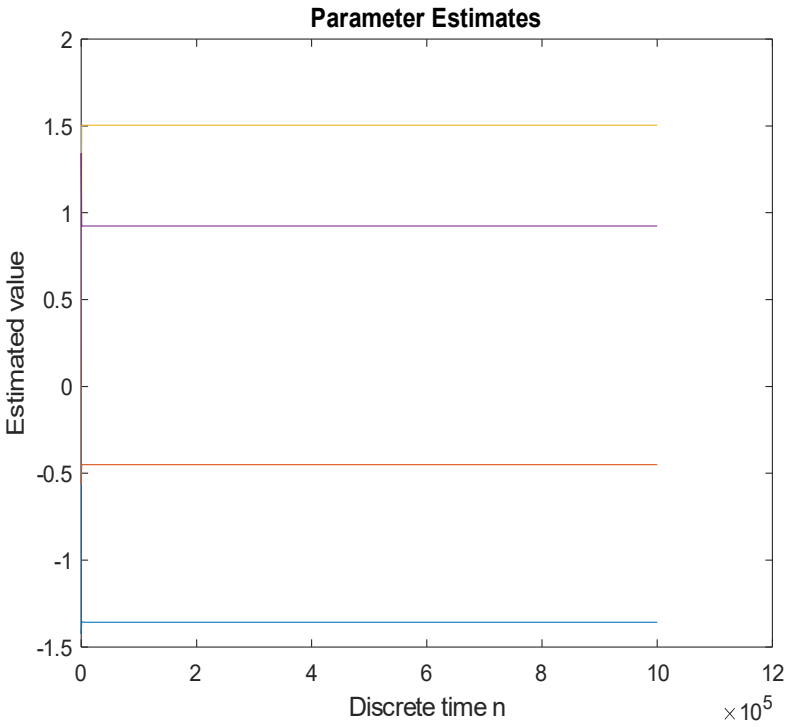

Figure 1C: RLS Parameter Estimates Over 1E6 Samples

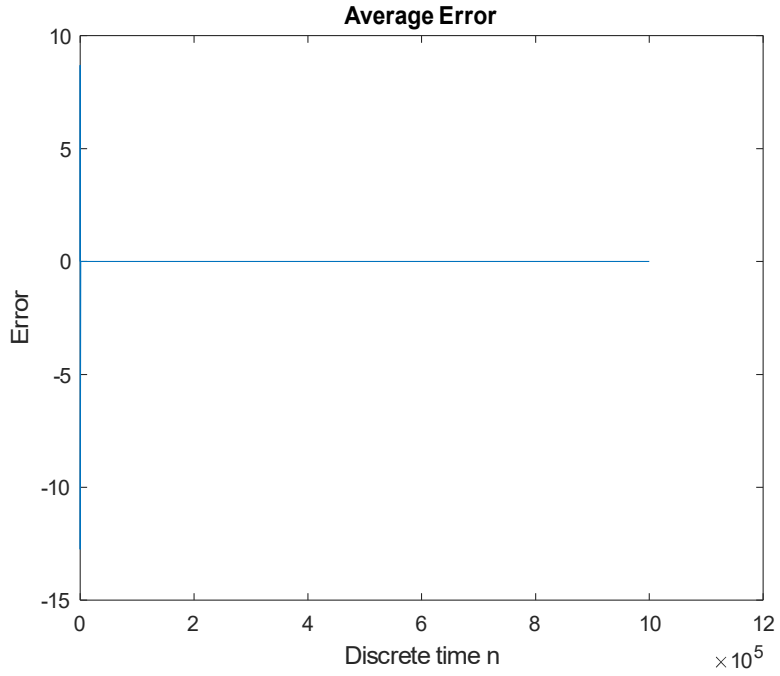

Figure 1D: RLS Error in Parameter Estimates

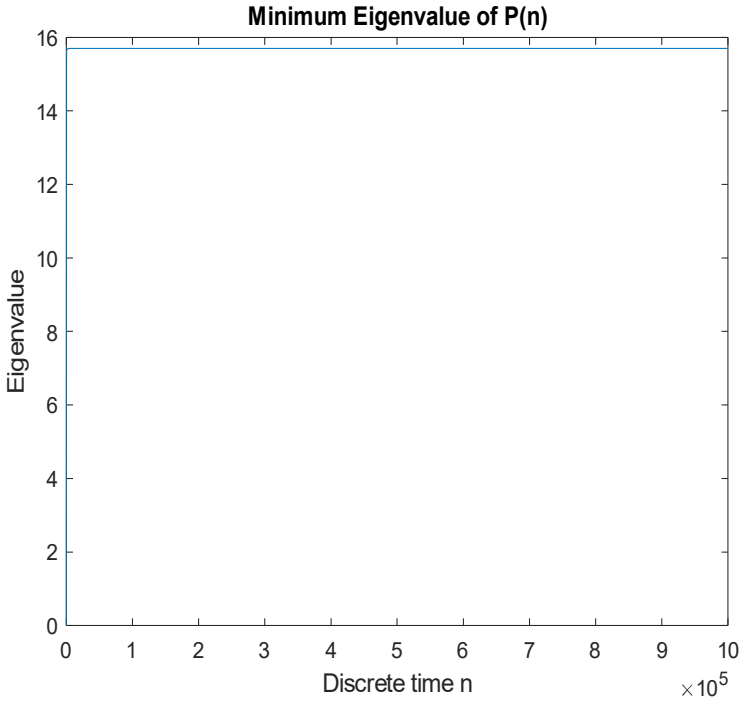

Figure 1E: Eigenvalue Convergence for RLS Linear Space Minimization

The above plots correspond to SIP part A, signal 1 (Appendix A). For this exercise, both NGA and RLS converged to the expected parameter sets for each control input signal. However, the plotted estimates are not quite there, as this convergence is very, very slow. Even after one million samples, the convergence is still not completely reached. In order to see that these values do indeed converge, the gain on the input signals can be raised in order to increase the concavity of the functional being controlled. This can be seen below for SIP part A, signal 1, where the gain on the input is increased 50 times.

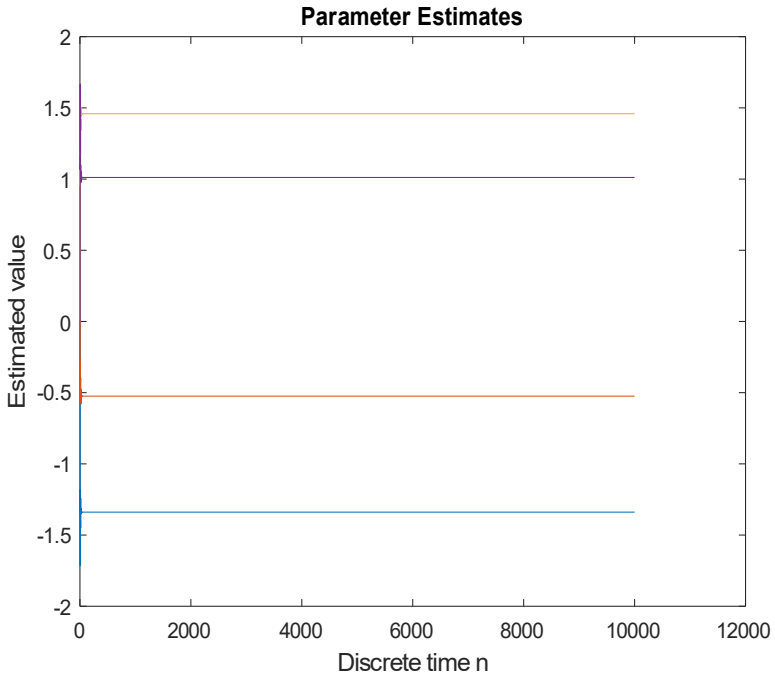

Figure 1F: NGA Parameter Estimates Over 1E4 Samples (Input Gain 50x)

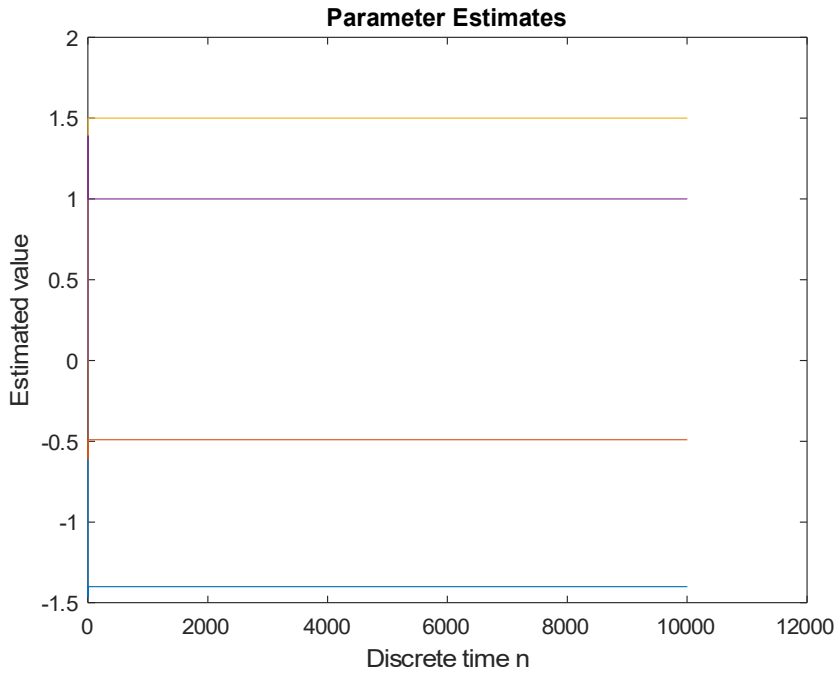

Figure 1G: RLS Parameter Estimates Over 1E4 Samples (Input Gain 50x)

These plots give a clear example of the differences discussed between NGA and RLS. From the above, it is obvious convergence occurs to the parameter set defined in SIP1, but there is slight variation in the NGA as opposed to the RLS as they compare to the true values. NGA, while extremely close, does not quite converge exactly, where as RLS is much, much closer. This is due to the fact that RLS has memory, as discussed in the summary of this section.

In the case for SIP, part 2 (SIP2), the control signal given repeats the same behavior as above, with the difference being the parameters do not converge to the expected values. Below is an example at a higher gain of the phenomenon.

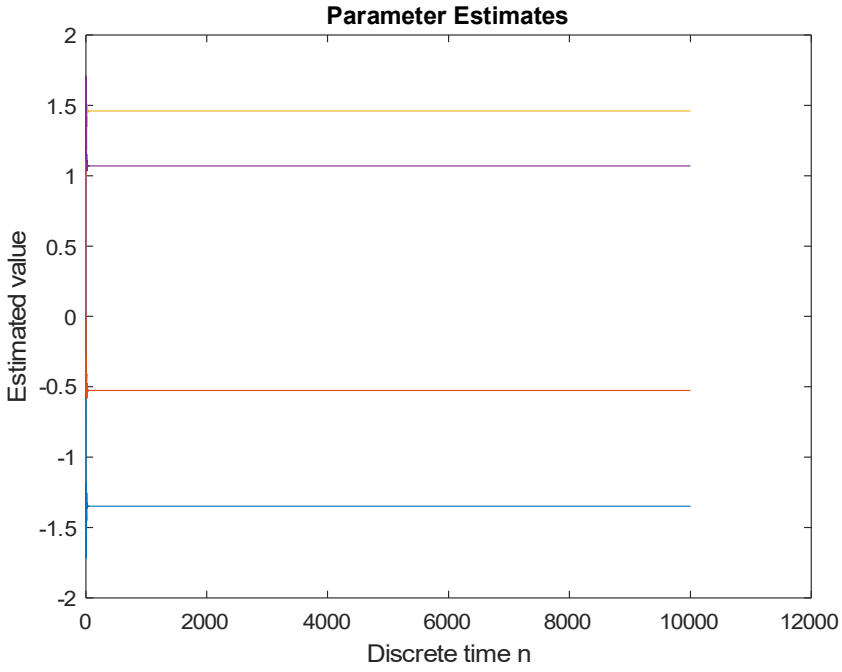

Figure 1H: NGA Parameter Estimates for SIP2

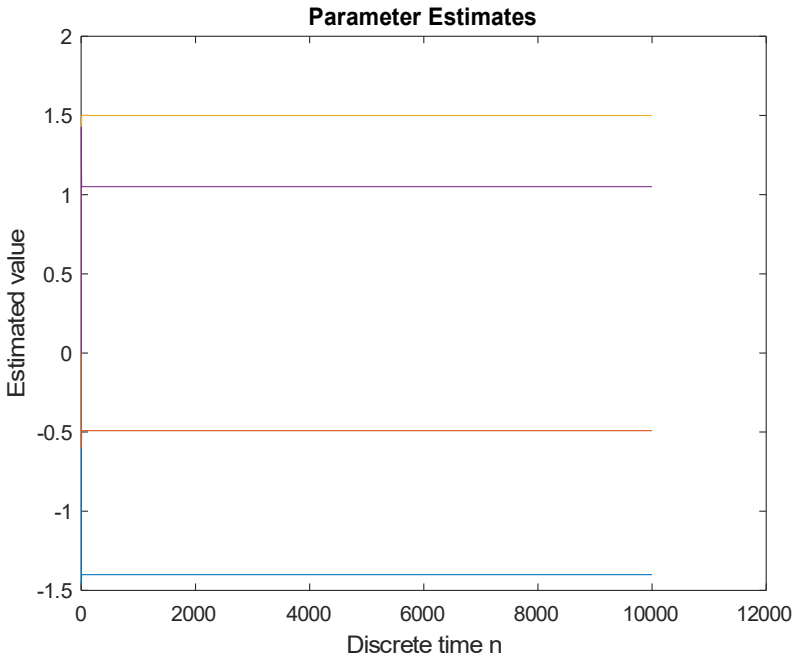

Figure 1I: RLS Parameter Estimates for SIP2

The values converged to are slightly different due to the fact that while the system is stable, it is a reducible control input. This means the transfer function associated with this control input loses a pole and the system is over-determined. By contrast, SIP, part 3 (SIP3) is under-determined and therefore has infinitely many solutions. Below are the plots for this system using input signal 1 and one million samples (shallow concavity case)

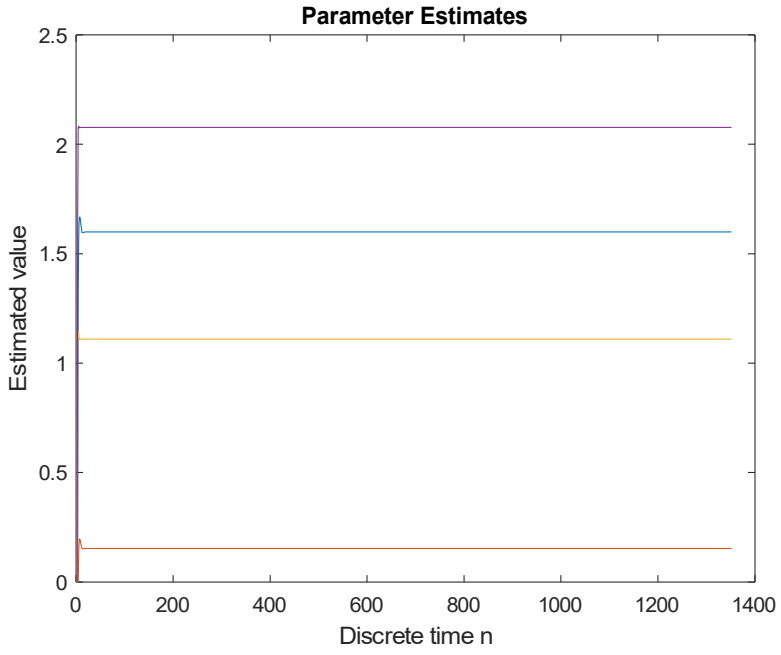

Figure 1J: NGA Parameter Estimates for SIP3, Control Signal 1

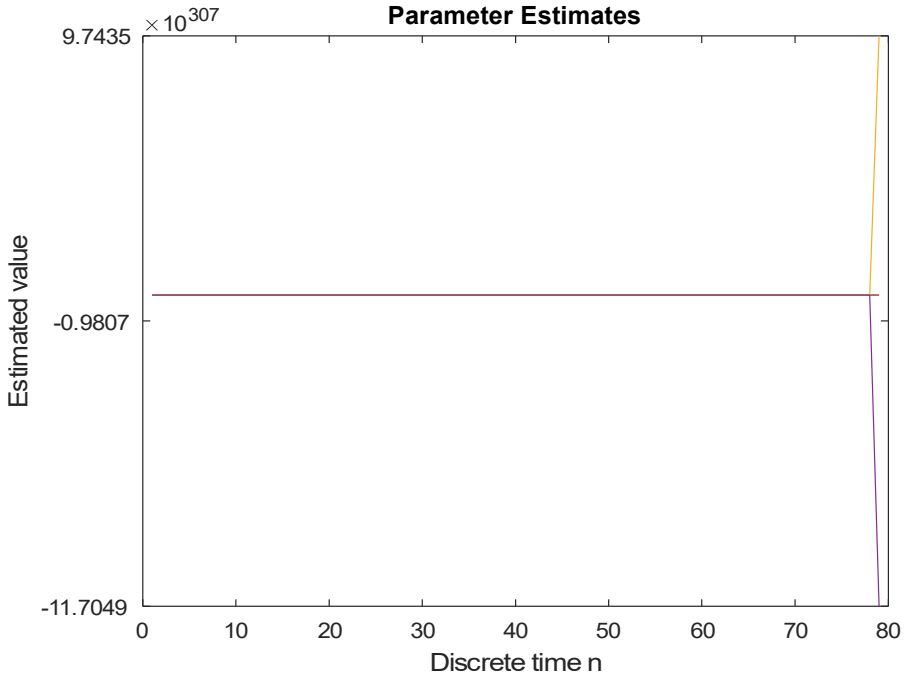

Figure 1K: RLS Parameter Estimates for SIP3, Control Signal 1

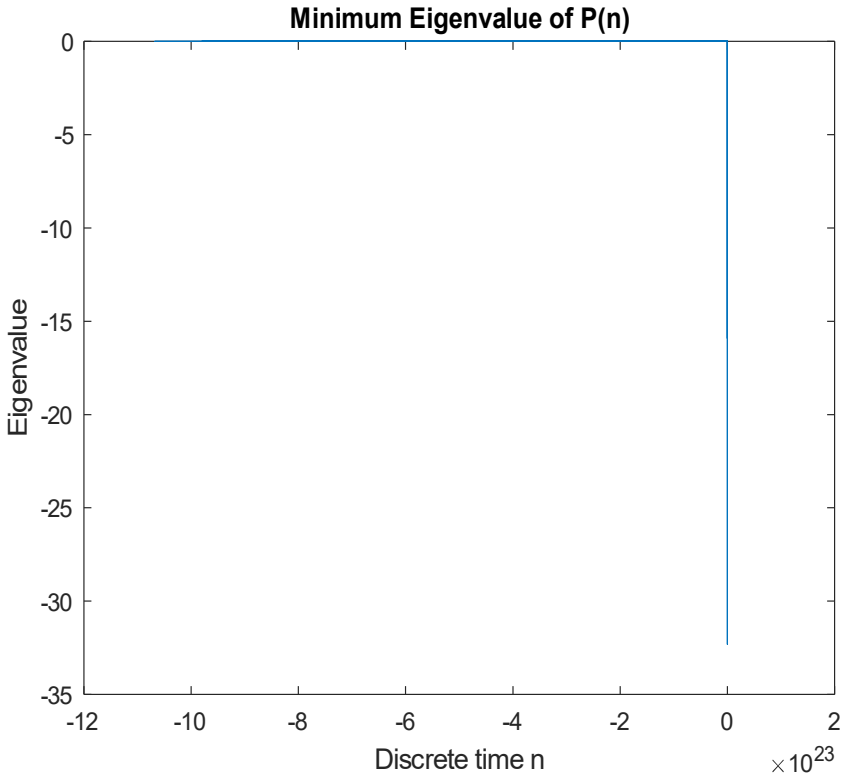

Figure 1L: Eigenvalue Convergence for RLS Linear Space Minimization (Control Signal 1)

It is obvious that these figures do not converge to the expected values and in fact, for the case of RLS, are completely unusable. Relating back to the summary for this section, these figures are an example of how the NGA can still converge and find stability at local minimums, even when the functional has no global minimum. The RLS algorithm fails

completely for a functional with no global minimum as the linear projector used to minimize the space is under-determined and therefore singular. Changing the gain on such a system has no effect on convergence or stability since the fundamental nature of the system is under-determined.

III. Distributed Self Tuning Consensus and Synchronization

A. Summary

The paper sharing the name of this section goes into great detail on achieving consensus between agents on a well connected graph. The general idea is that if each agent is aware of only its dynamics, and the dynamics of its neighbors, the entire system can and will converge to the same state. This paper utilizes the NGA discussed in Section II above.

B. **Report**

The purpose of this paper is to determine a method and algorithm to approximate convergence of multi agent systems. A good example of this is the flock behavior of birds, or the herd behavior of bison. Is it possible to model the phenomenon of multiple agent convergence, when each agent is in control of only their own dynamics, and capable of observing their nearest neighbors? If so, how can it be approximated? These are the question "Distributed self-tuning consensus and synchronization" attempts to answer.

To begin, it is important to identify the dynamic laws governing each agent. A simple model can be constructed with the following equations for each agent $i = \{1,N\}$:

$$\mathbf{x_i(t+1) = x_i(t) + v_i(t)}$$
$$\mathbf{v_i(t+1) = v_i(t) + \beta_i u_i(t)}$$

Clearly, these equations represent the position and velocity of each agent for a given time *t*. An interesting note on the latter of these two equations is that it is essentially a kinematic model for a driving force, where $\boldsymbol{\beta_i}$ is inversely proportional to the mass of the agent.

Each agent has control over its own kinematics, but for synchronization to occur, each agent must reach an average velocity, which is to be determined by the velocities of its nearest neighbors. This can be represented as an average of the other agents and from this, a velocity proportional to the driving force can be obtained:

$$\mathbf{v_{i\_avg}(t) = 1/(1+N_i)\ \Sigma\ v_j(t)}$$
$$\mathbf{\varphi_i(t) = v_{i\_avg}(t) - v_i(t)}$$

So now, the driver can be represented as:

$$\mathbf{u_i(t) = \theta_i(t)\varphi_i(t)}$$

with $\boldsymbol{\theta_i(t)}$ as the coupling parameter of the agent to the surrounding agents. This coupling parameter must have a value that is equal to the converging velocity for all agents in order to satisfy our initial goal. The full model for the velocity can now be expressed as:

$$\mathbf{v_i(t+1) = v_i(t) + \beta_i\theta_i(t)\varphi_i(t)}$$

From class, it can be seen this is a model resembling many of the adaptive systems that have been lectured on. The class examples differ in that they only dealt with a single system's convergence and not a single convergence for a set of like systems. More mathematics are required for this, specifically topics from graph theory and linear algebra.

For convergence of $\boldsymbol{\theta_i(t)}$, the coupling parameters, an algorithm utilizing the normalized gradient algorithm and a self-tuning consensus rule must be established. By examining the system model derived earlier, it is possible to form a matrix equation that relates the current agent as well as the neighboring agents:

$$\mathbf{v(t+1) = W(t)v(t)}$$
$$\mathbf{v(t)^T = [v_1(t), \ldots, v_N(t)]}$$

$$w_{ij}(t) = \begin{cases} \beta_i\theta_i(t) / (1 + N_i), & j \text{ is in the set } \eta_i \\ 1 - \beta_i\theta_i(t)\, N_i / (1 + N_i), & j = i \\ 0, & \text{otherwise} \end{cases}$$

where **W(t)** is a non-negative row stochastic matrix.

From class, and also for the purpose of this problem, the local cost function for the derivation of the NGA to be used is:

$$J_i(\theta_i) = .5(v_i(t + 1) - v_{i\_wa}(t + 1))^2$$

with $v_{i\_wa}(t + 1)$ as the one step delay weighted average of the current agent's velocity as well as that of its neighbors. In taking the gradient of such a cost function, one is left with the following expression:

$$\partial J_i(\theta_i)/\partial\theta_i(t) = (v_i(t + 1) - v_{i\_wa}(t + 1))\beta_i\varphi_i(t)$$

to which a recursive algorithm can be developed for the tuning of $\theta_i(t)$:

$$\theta_i(t + 1) = \theta_i(t) - \beta_i\varphi_i(t)e_i(t + 1)$$

$$\beta_i = (\mu_i/r_i(t))\text{sgn}(\beta_i)$$
$$e_i(t + 1) = v_i(t + 1) - v_{i\_wa}(t + 1)$$

where μ_i is the step size and greater than zero. From above, a new function $e_i(t + 1)$ is defined, where it represents the error between the velocity and the weighted average of velocities for the agent and its neighbors. The function $r_i(t)$ is also introduced and represents the normalization of the gradient, which in this case is equal to $1 + \varphi_i(t)^2$. Initial conditions for this algorithm are arbitrary so long as they are finite. This has the implication that regardless of initial states of each agent, the coupling parameters for each system should converge to common values. This can be further illustrated by recalling that $v_{i\_wa}(t + 1)$ is a weighted average. As time goes on, such that convergence is obtained, the majority of the weight for this average will be given to the agent's own velocity, thereby forcing the error function to zero. However, even with every system doing this with respect to their coupling parameters, the overall set of systems will only be driven to the common group velocity if their individual state velocity is bounded.

The next topic of discussion involves leaderless consensus, an important issue for system synchronization. It is a necessary topic to define as many real world synchronous systems are in fact leaderless and completely dependent on 'herd' behavior. There are three criteria that are mentioned in the paper “Distributed self-tuning consensus and synchronization” that will be summarized below.

One of the necessary criteria for the problem at hand to achieve leaderless consensus. is one which was already touched upon. It is that the velocity of the other agents with respect to the current agent are driven to zero. This can be described mathematically as:

$$\lim_{t \to \infty} (v_i(t) - v_j(t)) = 0$$

This however, is not the only necessity, as there are several functions which can achieve this, yet still not achieve convergence, or be bounded for that matter, for the system. This brings about the second criteria for leaderless consensus.

The second criteria is very much related to the first, in that the first criteria absolutely must hold true. In addition however, it must also hold that the current agent velocity converges to a consensus velocity as time approaches infinity. Mathematically, this can be written:

$$\lim_{t \to \infty} v_i(t) = v_c$$

This leads to the third point to be made about leaderless consensus. However, it is worth noting that for adaptive consensus, the positions of the agents will also converge to a finite distance only if the quantity $v_i(t) - v_j(t)$ converges toward zero quickly.

The third requirement for leaderless consensus to occur requires delving into the stability and convergence requirements of the self-tuner. This proves to be mathematically rigorous, so a brief, high level summary will be given instead.

Several assumptions must be made in order to show the self-tuning mechanism for the parameters is convergent. It must also be shown that the error and difference in agent velocity with the average velocity have finite total energies for all initial conditions (initial coupling parameters, positions, and velocities for each agent). The following assumptions will need to be made:

1. **The directed graph must be strongly connected.**
2. **The sign of β_i must be known and the upper bound of β_i must be defined and known.**
3. **$\theta_i(0)$ must be selected such that $0 < \theta_{0i}(0) < 2(1 - \alpha_i)/\beta_{i\_max}$, $.5 <= \alpha_i < 1$**

These three assumptions are summarized below.

The first assumption will be discussed briefly. The first assumption (1) simply means the group of agents are all connected directly to one another in the sense that nodes in a network are. There is correlation between the connecting nodes.

The second and third assumptions relate back to our equation for **W(t)**, the matrix relating the current agent to surrounding agents. The proof for (2) requires (3) and shows that this matrix is row stochastic and has a single eigenvalue equal to one. Being a row stochastic matrix means that by summing across any row in this matrix, that sum is equal to one. In other words, the sum of probability across all states for any given agent is one. The eigenvalue being equal to one means that in addition to being a row stochastic matrix, the matrix is also stationary, meaning it is time-invariant. Following through rigorous proof and utilizing the properties of **W**, it can be shown that the assumptions made with (2) and (3) are valid and the third requirement for leaderless consensus is reached. On a side, but related note, an interesting observation to make in (3) is the weight given to the current agent, α_i. The bounds on this weight imply that the majority of the weight must be given to the current agent, with at least one neighbor sharing in the weighing. This can be seen by the bounds given to α_i. The greater than or equal to .5 implies majority weighing, while the upper bound (less than one) implies that all importance can not be given to the agent (which makes sense b/c then it would be a set consisting of one system).

Although this paper is merely a summary of "Distributed self-tuning consensus and synchronization", it hopefully answered the opening question as well as gave insight into how consensus is achieved. Much of the mathematical rigor was passed over, but the major points to which this process requires were touched upon. This paper was extremely interesting and insightful.

Analysis

This section duplicates the simulations performed in the above paper, but with a different well connected graph **A:**

```
A(1,:) = [0 0 0 0 0 1];
A(2,:) = [0 0 0 0 1 0];
A(3,:) = [0 1 0 0 0 0];
A(4,:) = [1 0 0 0 0 0];
A(5,:) = [0 0 0 1 0 0];
A(6,:) = [0 0 1 0 0 0];
```

It is important to note that this graph is such that each agent is aware of only a single neighbor and that communication to an agent's furthest neighbor will take at most five hops. Intuitively, it is expected that the overall state consensus will result in a slightly longer transient, as consensus information will take longer to reach all agents. Below are plots for comparison.

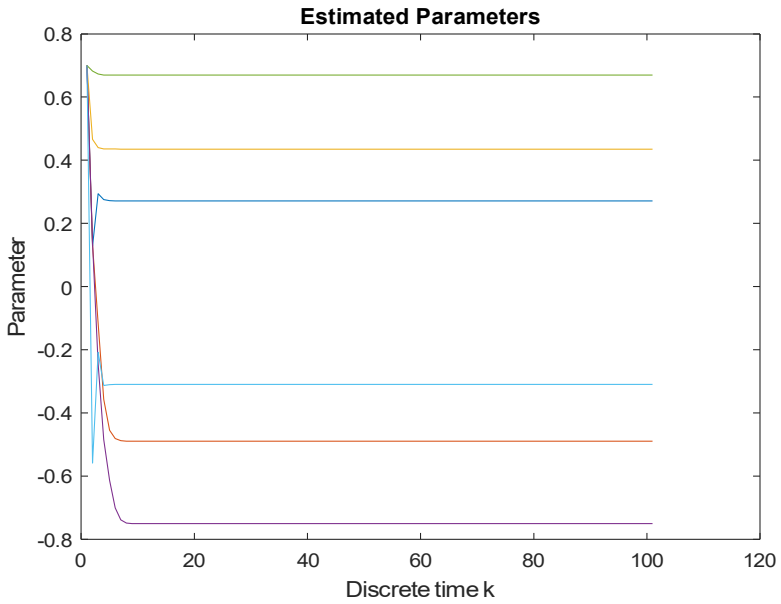

Figure 2A: Official Parameter Results Presented in Paper

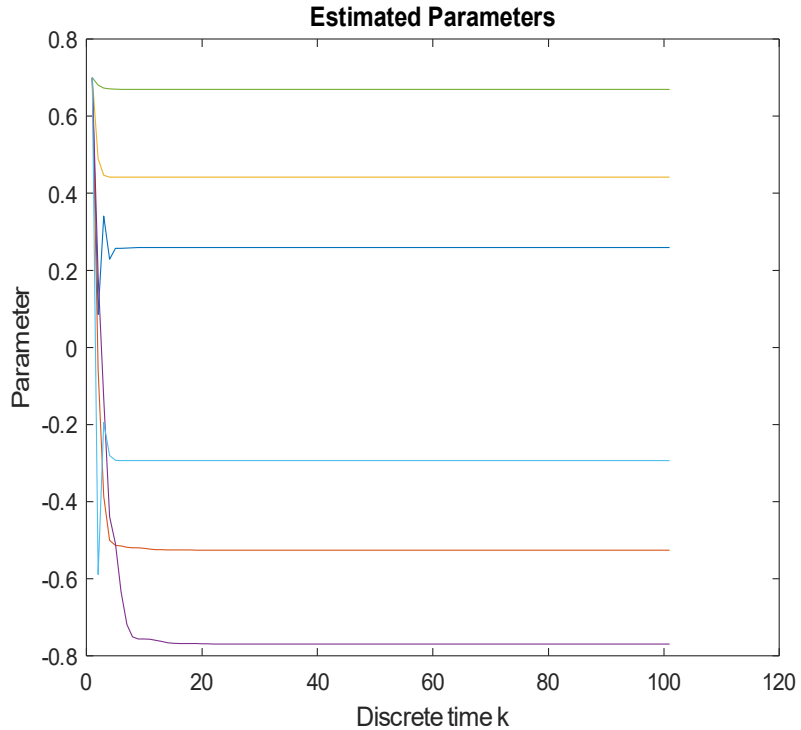

Figure 2B: Parameter Results Using Graph A

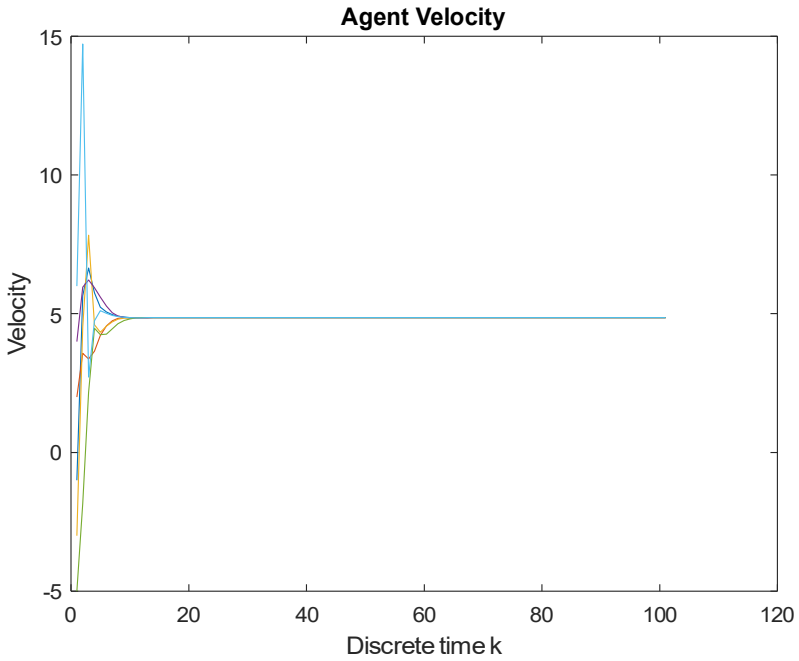

Figure 2C: Official Velocity Results Presented in Paper

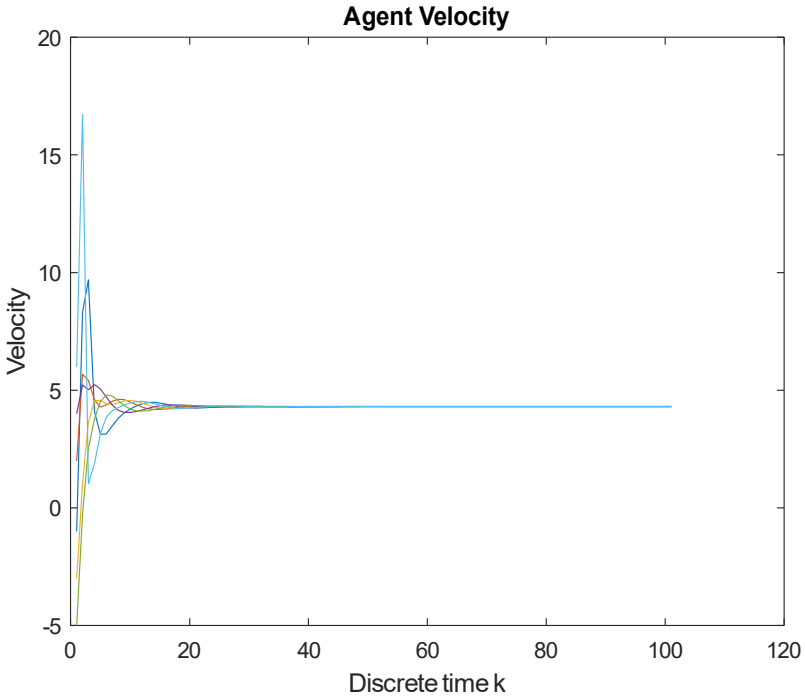

Figure 2D: Velocity Results Using Graph A

Figure 2E: Official Delta Position Results Presented in Paper

Figure 2F: Delta Position Results Using Graph A

As to be expected, the state information changed when the graph representing the agent communication changed. The overall control parameters saw very little change outside their respective gains, but even these gains remained relatively similar. The largest impact that was had, was on the individual agent velocities and on the distance between agents. This represents that the rate at which each agent is able to learn of every other agent's state information directly impacts the relative distance between agents, and the speed at which each agent converges to their respective position.

IV. Self-Tuning Consensus on Directed Graphs in Case of Unknown Control Direction

A. Summary

This paper primarily discusses a distributed algorithm for leaderless coordination in networked systems where each agent has an unknown control direction, yet all agents converge to a the same state.

B. Report

The goal of this paper was to develop an algorithm that takes dynamic systems with unknown initial dynamics and create a controller that steered all individual agents to a single group dynamic state. In order to do so, the authors assumed each agent was able to obtain information from any other agent, whether directly, or through one if its neighboring agents. An additional assumption made was that the agent's local input control was proportional to the error between the agent's state and the consensus solution. With these assumptions, the algorithm is designed to tune the inter-agent coupling parameters such that all agents converge to the synchronous state and the error driving each local controller converges toward zero.

Assuming the set of agents is identical and follow the given dynamics:

$$\mathbf{x_i(t + 1) = x_i(t) + bu_i(t),\ \ b=/0,\ i = 1, ..., N}$$

and t is a discrete time index, **x** and **u** are real valued numbers for the state and control signals, and the initial conditions of such an equation are finite values. For the sake of these N governing dynamics systems, it is assumed that they are only existent in forward time steps, and that for all times $k < 0$, the states and control signals are zero.

The agents are networked such that the network can be modeled by a strongly connected graph. Each node on the graph represents an agent, and each edge of the graph represents a connection between agents. A connection between agents belongs to the set of edges and not every pair of agents belongs to this set. This implies that in order for some agents to receive information from other agents, n hops must be made through the set of edges before the current agent can update. This is equivalent to t+n iterations in time. Additionally, the system dynamics of the above model can be represented as such:

$$\mathbf{m dx_i(\tau)/d\tau = u_i(\tau)}$$

where m is the mass of the individual agent, **x** is the velocity and **u** is proportional to a driving force. For the sake of simplicity, each agent i has the same mass as every other agent. Again, it is important to reiterate the only way each of these agents will converge to a synchronous state with all other agents is if the error between the current agent's state and all neighboring agent states converges toward zero. This process relates directly to the inter-agent gain b.

For cases where the sign of b is known, the algorithm of choice is the NGA (Section I). At each time step, each agent will run its own NGA for that step and transmit the values to the direct neighbors. The direct neighbors will use that information on the next time step to adapt their respective control gains to drive the error between the average state and their state toward zero. With only direct neighbors requiring updates, this controller is a first order control system. If nearest and next nearest neighbors had memory of these transactions, and the graph was not strongly connected, the order of the control system would be greater than one (unless the node was standalone, at which point it would never come to consensus with any other node).

This is a very brief report on this material, but there is a lot of overlap in Section III, with the primary driver for this section being the unknown directional controller governing the system dynamics for a known direction of the gain.

Analysis

Utilizing the controller outlined in the paper for Section IV, the following plots depict what the state convergence looks like for Gaussian randomly generated initial conditions [0,1] and the connected graph **A**:

```
A(1,:) = [0 1 1 0 1 0];
A(2,:) = [1 0 0 1 0 1];
A(3,:) = [1 0 0 0 1 1];
A(4,:) = [0 1 1 0 0 1];
A(5,:) = [1 0 1 0 0 1];
A(6,:) = [0 1 0 1 1 0];
```

An additional benefit to the code for this analysis is the ability to randomly generate a binary matrix **A** for further analysis (not done here). See Appendix B, problem3.m for more information.

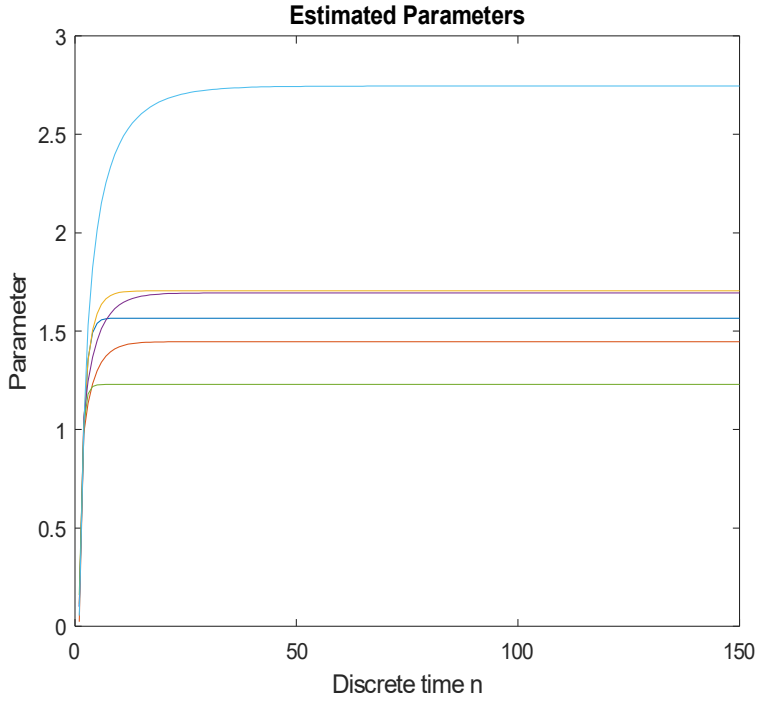

Figure 3A: Estimated Control Parameters for Graph A

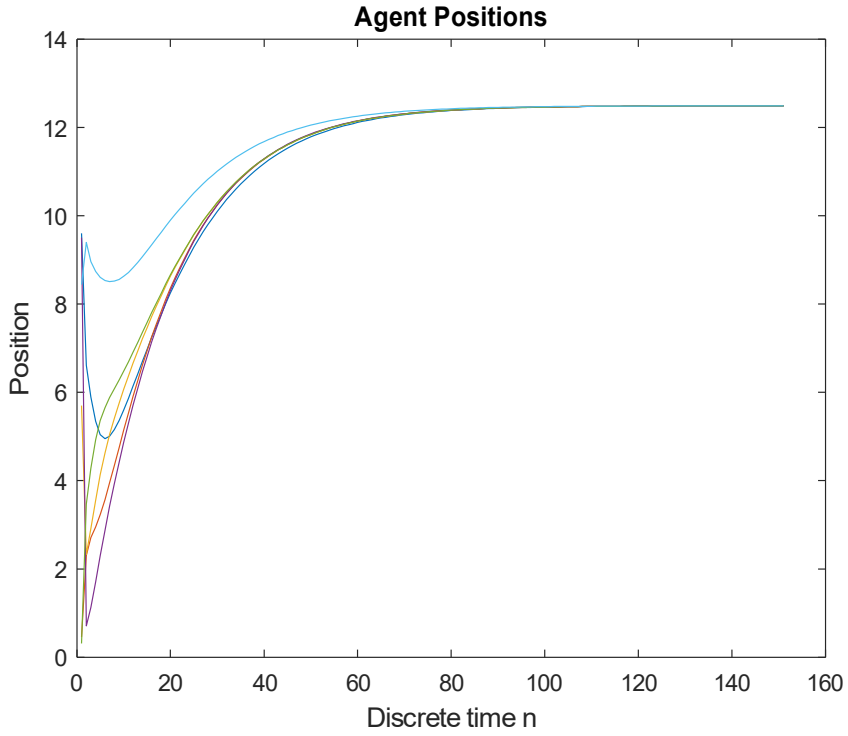

Figure 3B: Demonstrated State Consensus for Graph A

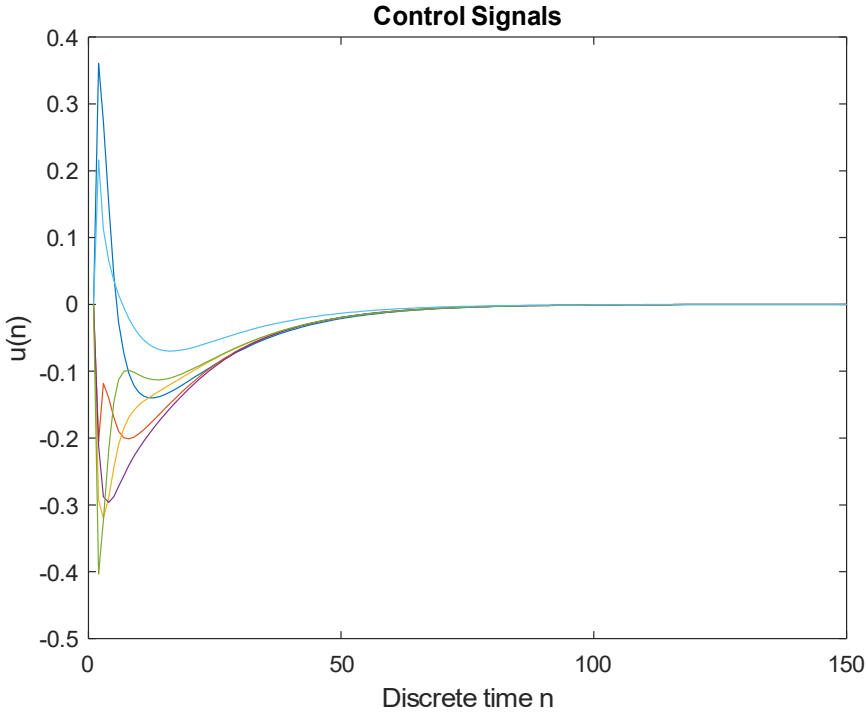

Figure 3C: Demonstrated Control Signal Convergence for Graph A

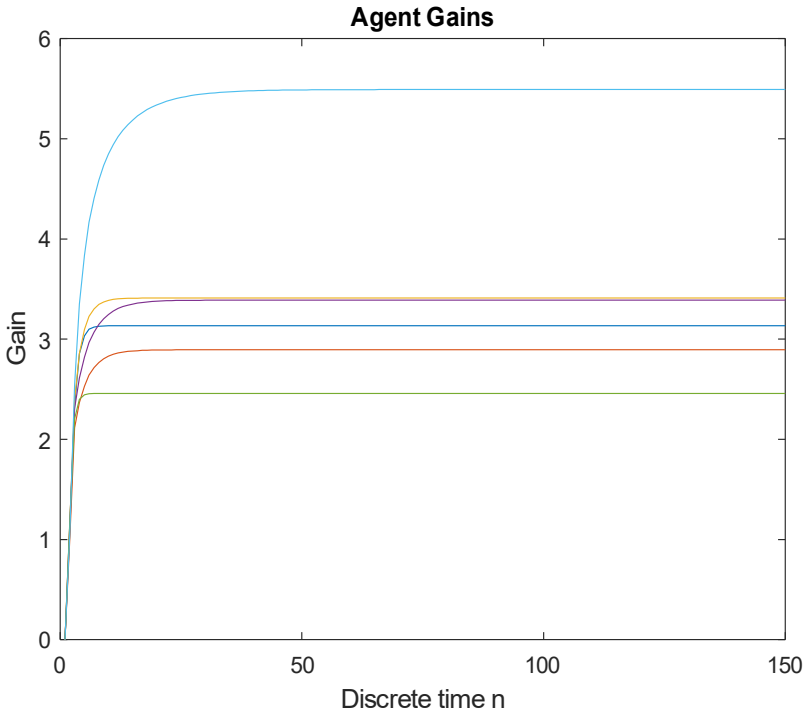

Figure 3D: Agent Gains (sgn(b) is known) for Graph A

As outlined in the report, it can be seen that convergence in **x** and **u** is met, and the agent parameters reach convergence with the gain b as the driver.

Appendix B

Link to code: https://github.com/jkootsher/MultivariateConsensus

References

M Radenkovic, M Golkowski, “Distributed self-tuning consensus and synchronization”, Systems & Control Letters, 76 (2015), pp. 66-73

M Radenkovic, M Tadi, “Self-tuning consensus on directed graphs in case of unknown control directions”, SIAM Journal on Control and Optimization, vol.54, pp. 2339-2353, 2016