CMDA 3605 Final Project Spring 2024

Anthony Merlin

Contents

1 Introduction **1**

2 Interaction Networks **2**

2.1 Static networks 2

2.2 Random networks 4

3 Averaging Algorithms **5**

3.1 Averaging over static networks 5

3.2 Averaging over switching networks 12

1 Introduction

Hello, this is officially my first Mathematical report, and now let's move on to the background of this project, the topic of this math report is based around the idea of Consensus, which in simple terms you could think of as agreement, one example that was used in the instructions that I liked was the idea of flocking birds on how they must agree on a certain direction as to where they would go.

2 Interaction Networks

2.1 Static networks

In this part we start off by being given a .mat containing 3 separate adjacency matrices called A1, A2, and A3 the matrices are shown as such

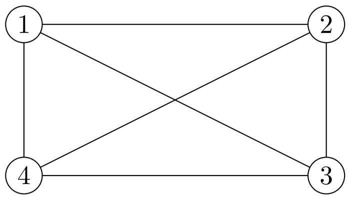
$$A1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$A2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

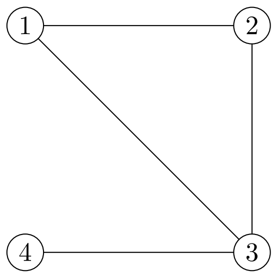
$$A3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

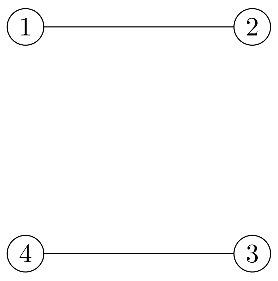
Now with that in mind let's draw our networks given by the three adjacency matrices which I have drawn out as such

$A_1 =$



$A_2 =$



$A_3 =$ 

After showing the networks let us now show the Laplacian and each of its eigenvalues which we do as such as

$$D - A = L$$

(and A is A1, A2 and A3)
Thus, our Laplacian is given as such (also in the octave code):

$$L1 = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

$$L2 = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

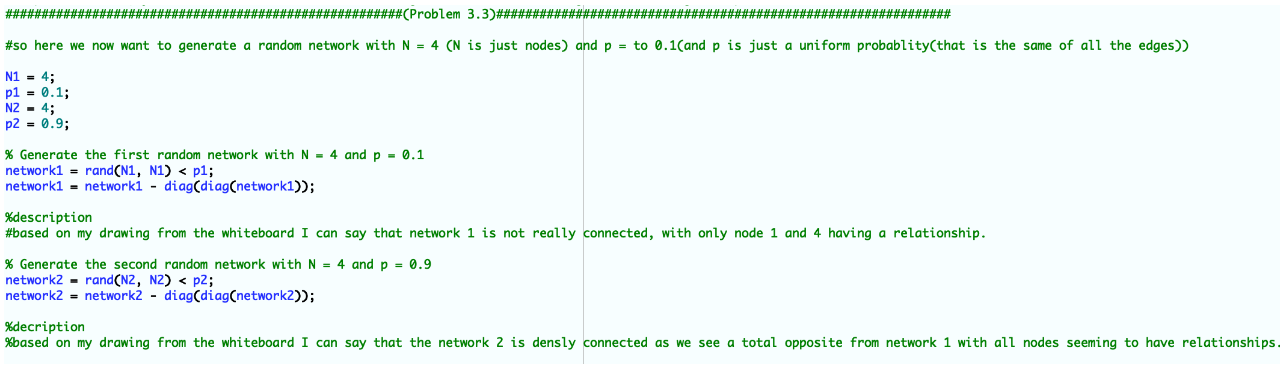
$$L3 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

And great, now we have our Laplacian matrices, let's look at the eigenvalues, and in particular look for eigenvalues of 0 and the structure of its eigenvectors. To start we can first look at L1 which has $\lambda = 4$, and 0 on which the eigenvector corresponding with the 0 eigenvalue is $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ as for L2 we get $\lambda = 0, 1, 4, 3$ and as for the eigenvector corresponding with the 0 eigenvalue is $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ and lastly, for L3 we get $\lambda = 0, 2$ and for the eigenvector corresponding with the 0 vector is

$\begin{bmatrix}1\\1\\0\\0\end{bmatrix} \begin{bmatrix}0\\0\\1\\1\end{bmatrix}$ Now that we know the eigenvectors and eigenvalues let's take a moment to see, aside from what the eigenvalue 0 does geometrically in a transformation sense, we can see that the eigenvalue 0 is being shared in L1 L2 L3 as well as the number of eigenvectors associated with 0 eigenvalue tells us the structure of the network we are working with for example in L1 the 0 eigenvalue had 1 eigenvector associating with a complete network, yet in L2 we can see there are two eigenvectors for the 0 eigenvalue showing 2 complete networks.

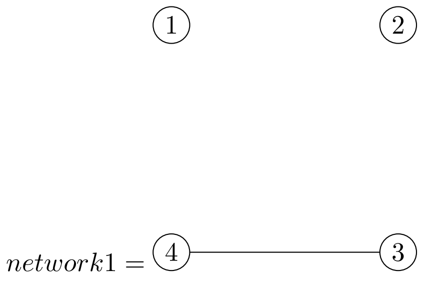
2.2 Random networks

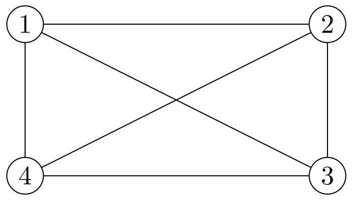
In this section, we are asked to write a script in order to write a random network with N = 4 and – 0.1 as well as N = 4 and p = 0.9. I do this part using octave code and here is the code I used for this part,



```
#####################################################(Problem 3.3)#############################################################

#so here we now want to generate a random network with N = 4 (N is just nodes) and p = to 0.1(and p is just a uniform probablity(that is the same of all the edges))

N1 = 4;
p1 = 0.1;
N2 = 4;
p2 = 0.9;

% Generate the first random network with N = 4 and p = 0.1
network1 = rand(N1, N1) < p1;
network1 = network1 - diag(diag(network1));

%description
#based on my drawing from the whiteboard I can say that network 1 is not really connected, with only node 1 and 4 having a relationship.

% Generate the second random network with N = 4 and p = 0.9
network2 = rand(N2, N2) < p2;
network2 = network2 - diag(diag(network2));

%decription
%based on my drawing from the whiteboard I can say that the network 2 is densly connected as we see a total opposite from network 1 with all nodes seeming to have relationships.
```

Based on the code here is a sketch of the networks being produced after one run, keep in mind it changes after each run.



$network2 =$ 

And based off these networks I can say that the network 2 is densely connected as we see a total opposite from network 1 with not as many connections thus it seems as if network 2 had an eigenvalue 0 it would have 1 eigenvector corresponded to it rather than network 1 if it had an eigenvalue of 0 then it would have most likely 2 eigenvectors giving the Structures of the networks.

3 Averaging Algorithms

3.1 Averaging over static networks

In this part we take a look at consensus through averaging algorithms while continuous and discrete. We then look into the idea of Equilibria and how it falls into play, on which we notice that models need to be designed so that the consensus state is an equilibrium. With that we can take a look at our first problem for this section, We are asked to explain why the vector $\alpha 1_4$ is an equilibrium for the discrete model

$$x(k+1) = (I_4 - \epsilon L)x(k)$$

and the continuous model

$$\dot{x} = -Lx$$

, and that equilibrium is true when all of the Adjacency matrices are considered A1, A2, and A3, First, let's start with checking with the discrete formula which starts out as such

$$x(k+1) = (I_4 - \epsilon L)x(k)$$

Now focus on the right side of the equation

$$(I_4 - \epsilon L)(\alpha 1_4)$$

Distribute

$$(I_4 \alpha 1_4 - \epsilon L \alpha 1_4)$$

$$\alpha(I_4 1_4 - \epsilon L 1_4)$$

Remark that $I_4 1_4 = 1_4$ as an identity matrix times a vector of 1's is equal to a vector of 1's as well as $L1_4 = 0$ (zero vector) (and for L we are considering the Laplacian for A1, A2, A3), thus based off that

$$\alpha(1_4 - \epsilon(0))$$

thus starts and stays

$$\alpha 1_4 = \alpha 1_4$$

Great now that, that works for the discrete model let try our Continuous Model Where we can start with this

$$\dot{x} = -Lx$$

and of course, lets focus on the right side of the eq

$$-L(\alpha 1_4)$$

Now switch

$$-\alpha(L1_4)$$

And remember from before that $L1_4 = 0$, thus

$$-\alpha(0) = 0$$

Thus, when the system is in state $x = \alpha 1_4$ The derivative $\dot{x} = 0$, meaning the state does not change over time. As for whether there are any other equilibriums for any of the three networks A1, A2, and A3, we can tell based on the eigenvectors associated with the eigenvalues of 0 from the Laplacian earlier, we see that L1 there is only one eigenvector for eigenvalue 0, thus no other equilibrium for this network and L2 we also see only one eigenvector for the eigenvalue 0, thus no other equilibrium. and lastly for L3 we see there are two eigenvectors for eigenvalue 0 thus, other than $\alpha 1_4$ there is another equilibrium.
Now after that, we can take a closer look into the ideas of eigenvalues and stability, in our project, and for a continuous time model, convergence to consensus can depend on the interaction network and initial conditions, but for discrete it's a little different as it also has the averaging weight ϵ which controls the stability for the equilibrium $\alpha 1_4$
And now based on that idea of stability that brings us to question 4.2 where we are asked to show a relationship between the $I_4 - \epsilon L$ eigenvalues and the graph Laplacian's eigenvalues, on which we will then write and solve an inequality for the values of ϵ that will allow the system to reach consensus, and for each of the three networks we will compute the bounds for ϵ that ensure the system reaches consensus. Great, now to start with question 4.2 where we can begin to write our relation as such

$$Ax = \lambda x$$

(remark we want to make it look like this $I_4 - \epsilon L$)

$$Lv_i = \lambda_i v_i$$

Then, multiply both sides by $-\epsilon$ (scales the eigenvalue equation)

$$-\epsilon L v_i = -\epsilon \lambda_i v_i$$

Now let's add vi to both sides

$$v_i - \epsilon L v_i = v_i - \epsilon \lambda_i v_i$$

Note (Identity matrix times a vector is a vector)

$$I_4 v_i - \epsilon L v_i = v_i - \epsilon \lambda_i v_i$$

And lastly just factor

$$(I_4 - \epsilon L) v_i = (1 - \epsilon \lambda_i) v_i$$

Thus this shows that λ_i is an eigenvalue of L, then $(1 - \epsilon\lambda_i)$ is an eigenvalue for $(I_4 - \epsilon L)$, Now back to question 4.2 to understand where we are at, thus we know that

$$Ax = \lambda x$$

and

$$Lv_i = \lambda_i v_i$$

and newly

$$(I_4 - \epsilon L) v_i = (1 - \epsilon \lambda_i) v_i$$

Now we need to show that the eigenvalues (absolute value) of $I_4 - \epsilon L$ are < 1, and remember since we are given the eigenvalues for $I_4 - \epsilon L$ are $1 - \epsilon\lambda_i$, the conditions for Consensus are:

$$\begin{cases} |1 - \epsilon\lambda_i| < 1 & \text{for all } \lambda_i \text{ not equal to } 0 \\ p(1 - \epsilon\lambda_i) < 1 & \text{for all } \lambda_i \text{ not equal to } 0 \end{cases}$$

Now let's solve our inequality

$$|1 - \epsilon\lambda_i| < 1 \text{ for all } \lambda_i \geq 0$$

$$-1 < 1 - \epsilon\lambda_i < 1$$

Let's focus on the left side

$$-1 < 1 - \epsilon\lambda_i$$

$$-2 < -\epsilon\lambda_i$$

$$2 > \epsilon\lambda_i$$

$$\epsilon < 2/\lambda_i$$

for $\lambda_i > 0$ Great now let's solve the right side

$$1 - \epsilon\lambda_i < 1$$

$$-\epsilon\lambda_i < 0$$

$$\epsilon\lambda_i > 0$$

$$\epsilon > 0$$

for $\lambda_i > 0$
Now let's combine the right and left

$$0 < \epsilon < 2/\lambda_i$$

for $\lambda_i > 0$
and now extend the inequality to include all non-zero eigenvalues

$$0 < \epsilon < 2/max(\lambda_i)$$

Where λ_i does not equal 0
Now based on this let's check for each of the three networks

$$A1 = 0 < \epsilon < 2/4 = 0.5 \text{ Consensus reached } 0 < \epsilon < 0.5$$

$$A2 = 0 < \epsilon < 2/4 = 0.5 \text{ Consensus reached } 0 < \epsilon < 0.5$$

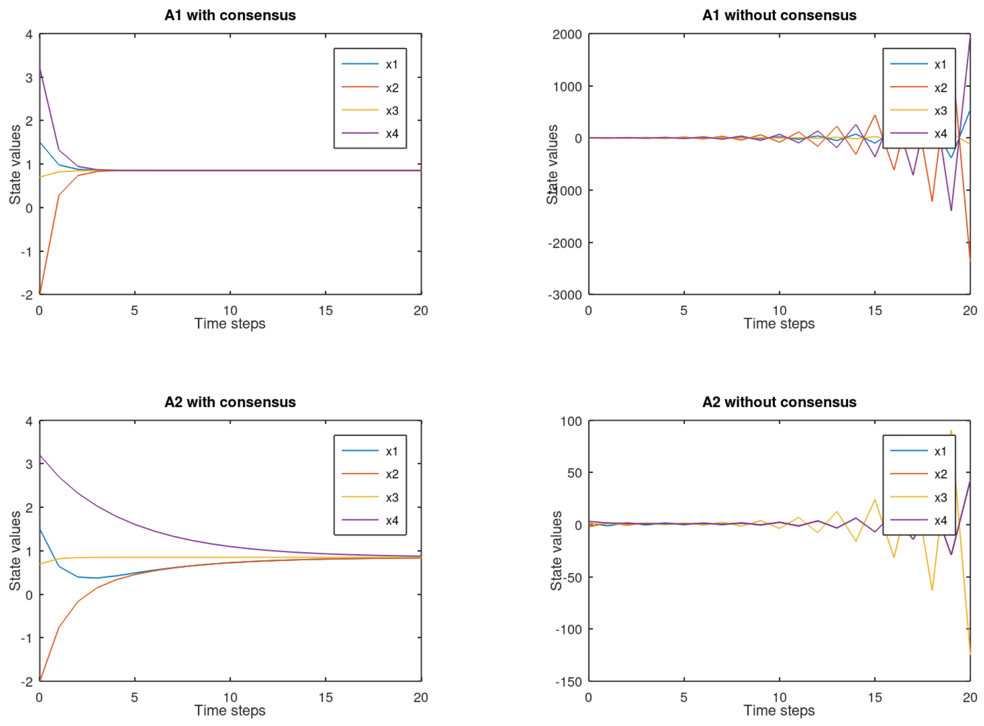
$$A3 = 0 < \epsilon < 2/2 = 1 \text{ Consensus reached } 0 < \epsilon < 1$$

Great we now have computed the bounds for ϵ that ensure the system reaches consensus.
The last topic we talk about in 4.1 results around the idea of simulations, as we have an idea as to what the networks and what values ϵ enable the network Individuals to reach consensus, let demonstrate this behavior through simulations. For the last question in part 4.1, we are asked to use Matlab to simulate the averaging algorithms to demonstrate the effect of the interaction network on consensus The first thing we want to do is fully make a .mat that contains a 4x1 vector that we will use as our initial condition, for our simulation

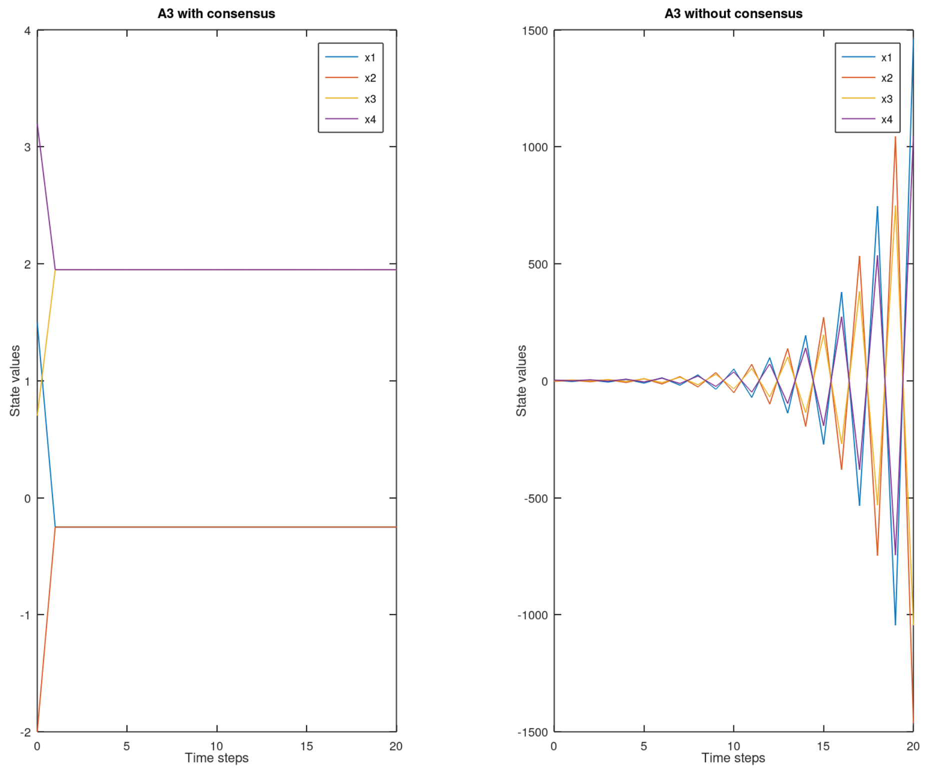
```
###########################################(Problem 4.3(1))##########################################################
% Here we have laoded our new initial_conditions vector

load('initial_conditions.mat');
disp(initial_conditions);
```

I decide to code this using octave and yes, I made a .mat file with octave. The vector I ended up using was $x_0 = \begin{bmatrix} 1.5000 \\ -2.000 \\ 0.7000 \\ 3.2000 \end{bmatrix}$ and above is the code I used to load this into our Project.m file so that we can use it.

After this, we come back to our simulation, and the next step needed is for us to simulate the discrete-time algorithm for 20-time steps for our interaction networks for A1 and A2, where we will then select two values for ϵ: one being too large for consensus and the other achieving consensus. On which we will then plot.

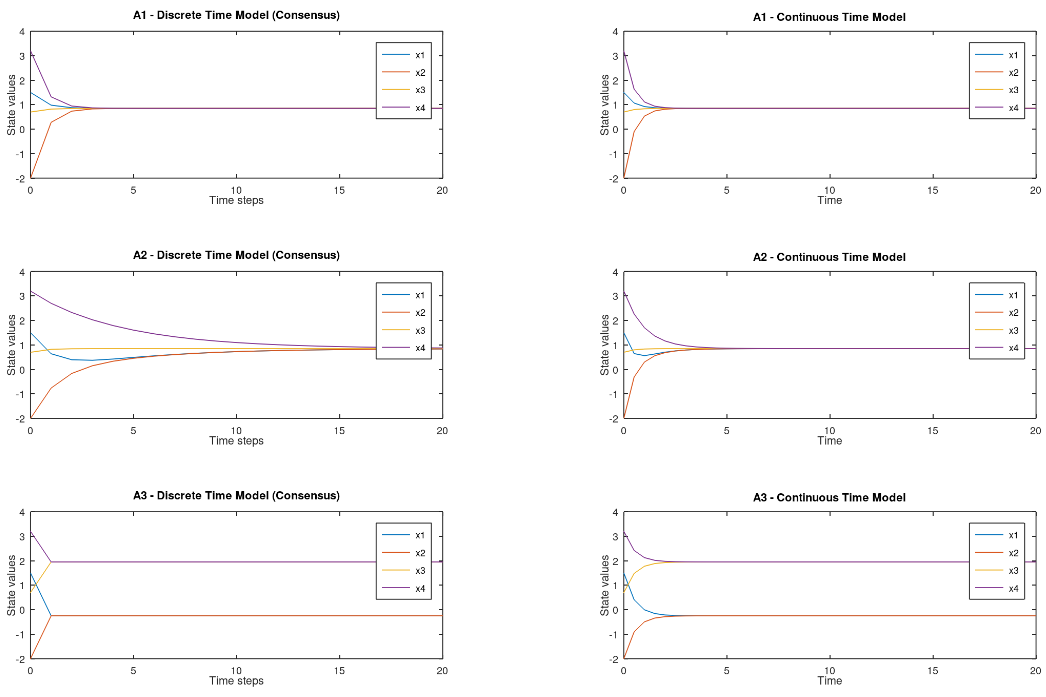


Based on the plots, we can clearly see the interaction networks A1 and A2 reach consensus when ϵ does as well as how our initial guesses eventually converge over the 20 time steps, and on the opposite end we can see for the interaction networks A1 and A2 that don't reach consensus, and how our initial guesses for those never converge and end up becoming much more static in the later time steps of the 20, and on one last note notice how each of these plots seem to correspond to one horizontal line, and that is due to the structure of the network as you maybe remember our eigenvalues from our Laplacian matrices and their eigenvectors associated with it.
As for the next part of the question we are now asked to simulate the discrete algorithm with A3 with two values of ϵ again one that reaches consensus and the other that doesn't reach, again I decided to use octave code for this part one which I received this plot

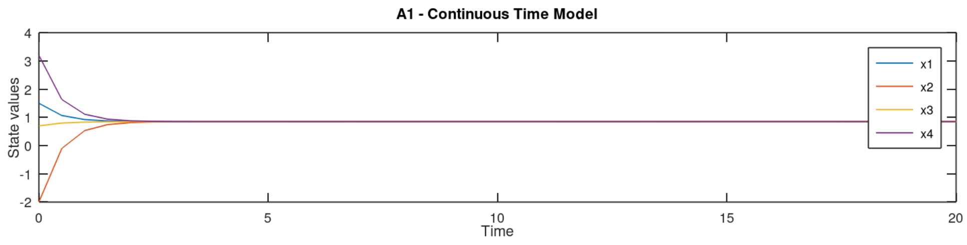


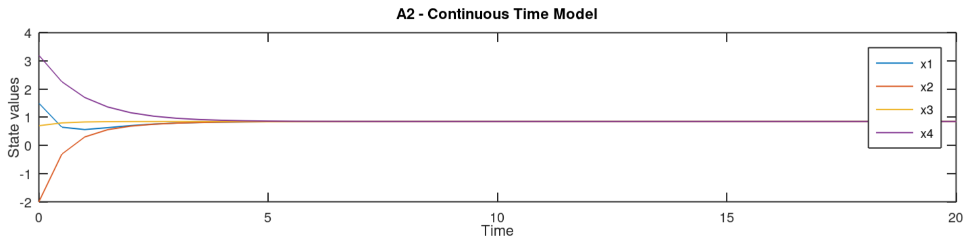
And as we can see based on the plot there is a major difference compared to A1 and A2 plots as there are two horizontal lines on which it converges, and

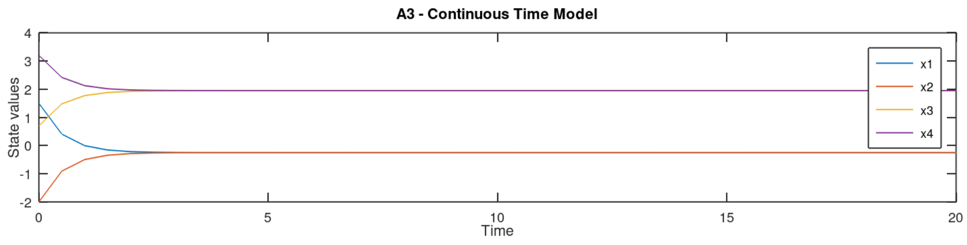
that is due to as we said above the structure of the and if you remember the eigenvalues of the Laplacian for A3 on which it had two eigenvectors for the associated 0 eigenvalue thus further proving this understanding of the structure of the network as well as we can see that for the A3 on which it does not converge to consensus where our initial guess never converge and end up blowing up in the later time steps.
Great, now for the last part of this question we are asked to use the Runge Kutta method that was talked about in Module 11 to simulate the continuous time model for the 20 units of time with a time step of 0.5, on which we will then simulate with A1, A2, A3, again I used octave for my code and here is my plot that demonstrate the difference between the continuous and discrete time model.



Thus based off the plots and our comparison plot with Discrete vs Continuous we can clearly see a difference of how it converges from the initial value x0 as well as the structure of the networks, thus finishing our section from 4.1, As well here a plot of just the continuous for a better visualization.



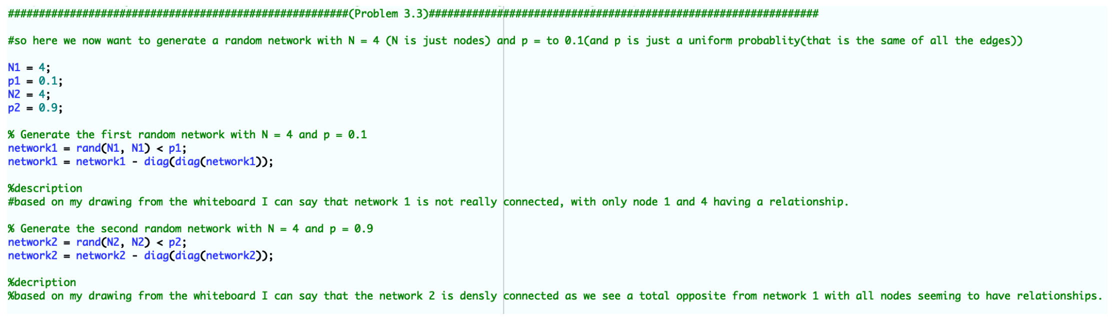




3.2 Averaging over switching networks

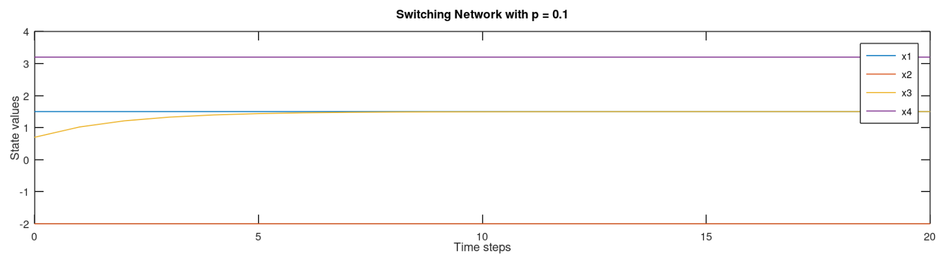
This section focuses on the idea of Averaging over switching networks, we begin by coming back to our interaction network, and we allow it to change in time using our random networks from section 3 (code for it shown below)

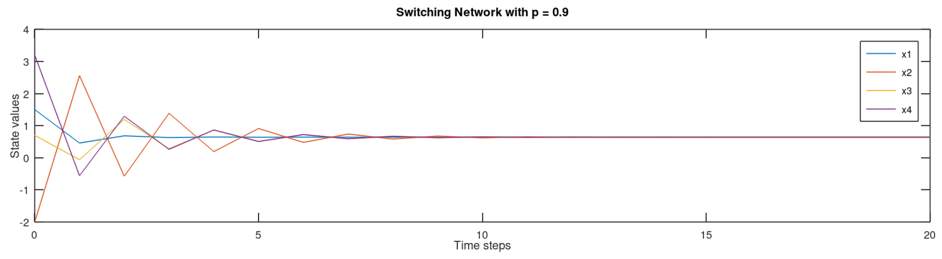
So, for the first part, we are asked to write a script that performs the discrete-time algorithm where the interactions network is an independent realization of a random network that switches at each time step. So for the first part, the code performs the discrete-time algorithm where the interaction network is an independent realization of a random network that switches at each time step. It does this by using two separate networks, network1 and network2, with different probabilities (p = 0.1 and p = 0.9), and simulating the algorithm for each network independently.
The second part of the problem then asks us to simulate the algorithm with a switch network using parameter values N = 4, ϵ = 0.4, and p = 0.1, 0.9 for 20 time steps. So the code then simulates the algorithm with switching networks



```
#######################################################(Problem 3.3)###############################################################

#so here we now want to generate a random network with N = 4 (N is just nodes) and p = to 0.1(and p is just a uniform probablity(that is the same of all the edges))

N1 = 4;
p1 = 0.1;
N2 = 4;
p2 = 0.9;

% Generate the first random network with N = 4 and p = 0.1
network1 = rand(N1, N1) < p1;
network1 = network1 - diag(diag(network1));

%description
#based on my drawing from the whiteboard I can say that network 1 is not really connected, with only node 1 and 4 having a relationship.

% Generate the second random network with N = 4 and p = 0.9
network2 = rand(N2, N2) < p2;
network2 = network2 - diag(diag(network2));

%decription
%based on my drawing from the whiteboard I can say that the network 2 is densly connected as we see a total opposite from network 1 with all nodes seeming to have relationships.
```

using the specified parameter values: N = 4, $\epsilon = 0.4$, and p = 0.1, 0.9 for 20 time steps. and the stochastic nature is due to, network1 and network2, as it changes after each run. One great way to showcase this was by plotting the results for each switching network shown below



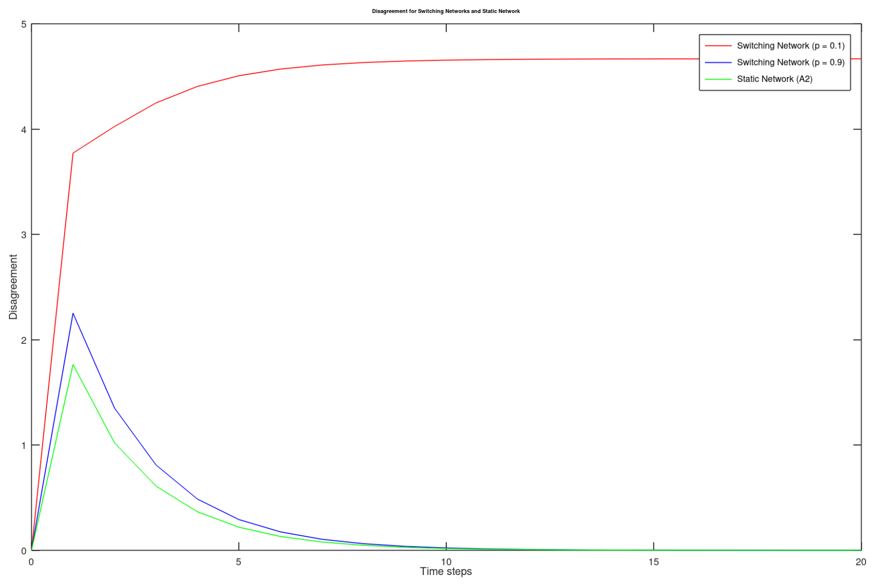


for further assurance, we can take a look at our network1 and network2 for this run which are

$$network1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$network2 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

and based on the plot we can also see that when p=0.1 it converges slower rather than when p=0.9, most likely due to the structure of the network.
For the last part of the project, we are asked to, compute the disagreement $\|\xi\|$ for the two simulations above (the one with p = 0.1 versus the one with p = 0.9) and for a static network that uses A2 and $\epsilon = 0.4$ for 20 time steps. Plot the disagreement for each simulation with time on the horizontal axis. Describe your observations. What is the effect of interaction networks that switch in time? so the code here computes the disagreement and plots it for the switching networks from earlier and the static network (A2) for 20 time steps, here is the plot:



and now based on the plot, the switching network with p = 0.1 starts with a higher initial disagreement compared to the other two cases, as well as the switching network with p = 0.9 has a lower initial disagreement compared to the p = 0.1 case, and it converges faster to a low disagreement level. The static network A2 exhibits the lowest initial disagreement among the three cases, and in Regards to the interaction networks that switch in time, I can say that Switching networks with higher connection probabilities (p = 0.9) tend to have lower initial disagreement and faster convergence to consensus compared to networks with lower connection probabilities (p = 0.1).