COMP 590 Homework 3

Mark I. Edwards

April 21, 2016

Problem 1

I first prove that if A is a matrix with eigenvalue λ and eigenvector v, then λ^n is an eigenvalue of A^n for positive integers n.

I prove this via induction. Our base case is our initial assumption, so I proceed to the induction step. I suppose that

$$A^n v = \lambda^n v$$

And then show that

$$A^{n+1} v = \lambda^{n+1} v$$

From our assumption, I left-multiply both sides by A, yielding

$$AA^n v = A^{n+1} v = A\lambda^n v = \lambda^n Av$$

From our initial assumption, we know that

$$Av = \lambda v$$

thus

$$A^{n+1} v = \lambda^n (\lambda v) = \lambda^{n+1} v$$

as required.

With this in mind, I now turn to consider

$$\lim_{n\to\infty} \lambda^n$$

when

$$\lim_{n\to\infty} A^n$$

exists (A^n does not approach infinity). We know that $\lim_{n\to\infty} \lambda^n$ exists, by right-multiplying both sides by v and noting the similarity to what we just proved. Now let $\lambda = ae^{i\theta}$ for some real a and θ. There are now 3 possibilities:

$$|a| < 1$$
$$|a| = 1$$
$$|a| > 1$$

Now if $|a| < 1$, $\lambda^n \to 0$. If $|a| = 1$, then $ae^{in\theta} = \lambda^n$ will only converge if $\theta = 0$, since otherwise λ^n will continuously traverse the edge of the unit circle. If $|a| > 1$, $|\lambda^n| = |a|^n$ will grow infinitely as n increases, and so λ^n will not converge. In summary, λ will only converge if $|\lambda| < 1$ or $\lambda = \pm 1$.

As stated in class, $A = e^{-L(D)}$ contains a rooted out branching if and only if the agreement algorithm converges. As of such, we know that the eigenvalues A are either ± 1 or have magnitude less than 1, because the agreement algorithm can then be written as $x(t) = x_0 A^t$.

Problem 2

In order to analyze this, let us consider the discrete time agreement algorithm.

$$z(k+1) = e^{-\delta L(D)} z(k)$$

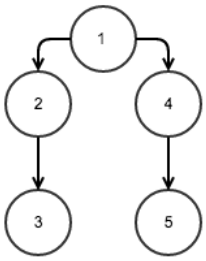
As noted on page 74 of Mesbahi and Egerstedt. And as noted by Corollary 4.2 of the same book, the digraph D is rooted out branching if and only if one of the columns of $e^{-\delta L(D)}$ is positive. From this, we can see that if the digraph D is rooted out branching,

$$V(z(k+1)) < V(z(k))$$

because each time step k updates the z value by taking a convex combination of itself and the other vertex values, and because one column of $e^{-\delta L(D)}$ is positive. This means that the maximum and minimum values approach one another since they are "connected" by that positive column, representing some connection within the digraph. This assumes that the minimum and maximum values are different, of course, but if they are the same then we are already in the agreement subspace.

I show this in effect for the directed graph in Figure 1. With adjacency

Figure 1: A 5 Node Directed Rooted Out Branching Graph



matrix

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

And degree matrix

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

Which yields the Laplacian

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

with initial values

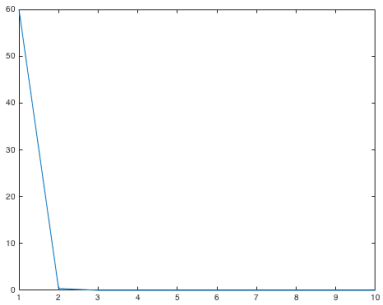
$$\begin{bmatrix} 50 \\ 30 \\ 45 \\ 70 \\ 90 \end{bmatrix}$$

I implement the algorithm in `p2.m` and get the Lyapunov function using

```
max(vals) - min(vals)
```

When plotted, this yields

Figure 2: Lyapunov Function



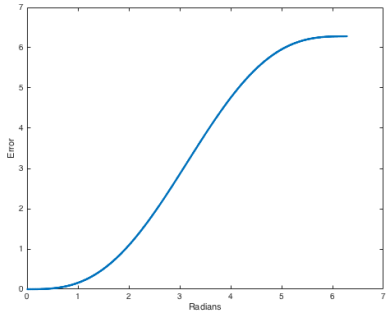
Problem 3

Before I address the key issue of this problem, I would like to bring up the small angle approximation which says that

$$\theta \approx \sin\theta$$

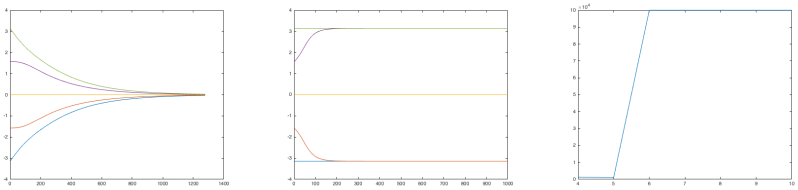
for small angles θ. This is very closely related to the issue of convergence of this algorithm, because the failure of this approximation for larger angles results in the failure of the algorithm when the differences between the values of neighboring vertices is sufficiently large. The following figures were generated

Figure 3: Absolute Error of the Small Angle Approximation



by varying values in the code in `hw3_problem3.m` which calls the function that implements the algorithm in `p3.m`. The code in `hw3_problem3.m` generates connected graphs with 5 nodes and gradually adds edges until the graph is fully connected. In the case that the algorithm does not converge, `p3.m` will automatically exit after 10^5 steps. All graphs have exactly 5 nodes, but varying numbers of edges.

Figure 4: Initial Values Between ($-\pi$ to π)



(a) A Path From 1 to 5

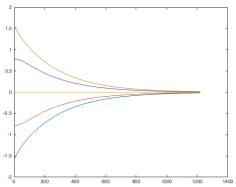
(b) A Fully Connected Graph

(c) Number of Steps vs Edges

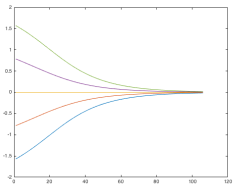
From Figure 4(b) we can see an example of an initial set up that does not converge. In fact, as shown in Figure 4(c), applying the algorithm to all of

the graphs in my progression (of sequentially adding edges) will diverge for all graphs with more than 5 edges. Furthermore, while impossible to see clearly due to the size of the figures, the graph with 5 edges converges faster than the one with 4.

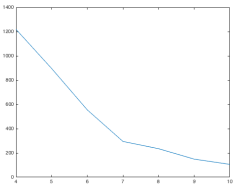
Figure 5: Initial Values Between $(-\pi/2$ to $\pi/2)$



(a) A Path From 1 to 5



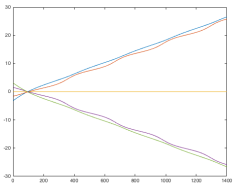
(b) A Fully Connected Graph



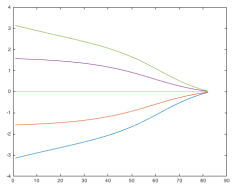
(c) Number of Steps vs Edges

However, when we limit the range of initial θ values to $-\pi/2$ to $\pi/2$, the algorithm converges quite quickly, as seen in Figure 5. From our Assignment 1, we found that the standard agreement protocol with similar initial values could take upward of a thousand iterations with similar step size on a fully connected graph, while this algorithm takes just over 100. Furthermore, now that every graph generated by the progression converges, we see the expected decrease in the number of required steps as the number of edges increases.

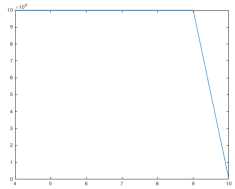
Figure 6: Initial Values Between $(-\pi$ to $\pi)$
Omega between $(\pi \cdot 0.9, -\pi \cdot 0.9)$



(a) A Path From 1 to 5



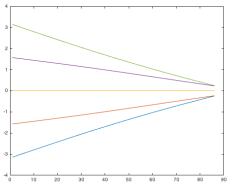
(b) A Fully Connected Graph



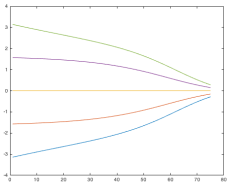
(c) Number of Steps vs Edges

Generally, random values of ω_i will not improve convergence. However, it is possible to use ω to try to improve convergence by "nudging" the values in the right direction. In Figure 6, we see the same set up as Figure 4 with the addition of an ω value that represents a bias representing prior knowledge of the average of the other points. This is a powerful method to try to achieve convergence with this algorithm when it would otherwise be impossible, but it certainly isn't infallible. In Figure 6(a) we that the values approach convergence, but then "miss their mark" and diverge. This can be adjusted for by relaxing the convergence criterion. In Figure 7, I relax the convergence criterion from a

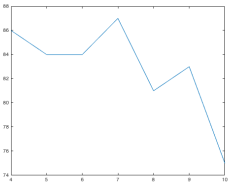
Figure 7: Initial Values Between $(-\pi$ to $\pi)$
Omega between $(\pi \cdot 0.9, -\pi \cdot 0.9)$
Convergence Criterion Relaxed



(a) A Path From 1 to 5



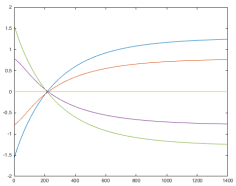
(b) A Fully Connected Graph



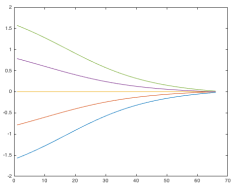
(c) Number of Steps vs Edges

1% difference from the mean to a 10% difference. This allows all the graphs in the progression to converge, and in Figure 7(c) we see the expected reduction in convergence steps with respect to increasing in connectivity, although the curve is now a jagged one. Generally, relaxing the convergence criterion is necessary if we use ω to try to improve convergence. In Figure 8 below, we see that adding an ω value to the setup from Figure 5 causes the algorithm to diverge for most graphs in the progression, following the pattern seen in Figure 8(a). Figure 8 shows that without relaxing the convergence condition, even graphs

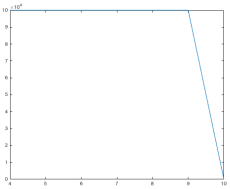
Figure 8: Initial Values Between $(-\pi/2$ to $\pi/2)$
Omega between $(\pi \cdot 0.3, -\pi \cdot 0.3)$



(a) A Path From 1 to 5



(b) A Fully Connected Graph



(c) Number of Steps vs Edges

that normally converge will diverge.

In summary, this algorithm attempts to trade reliability for convergence speed. Generally, convergence depends on the difference between the values of connected nodes. This can be "biased" using the ω values, but using this often requires that we relax our convergence criterion.