Homework #3

AA 597: Networked Dynamics Systems
Prof. Mehran Mesbahi
Due: Feb 2, 2024 11:59pm
Soowhan Yi

P1. How would one extend Exercise 3.6 to n particles in three dimensions?
First, random graphs with n nodes were created.

P2.

$$\dot{x}_i(t) = \sum_{j \in N(i)} (x_j(t) - x_i(t)), i = 1, \cdots, n$$

Consider vertex i in the context of the agreement protocol(3.1). Suppose that vertex i (the rebel) decides not to abide by the agreement protocol, and instead fixes its state to a constant value. Show that all vertices converge to the state of the rebel vertex when the graph is connected.
1. Prove that the graph contains rooted out branching 2. Jordan decomposition 3. prove $\lambda_1 = 0$, $\lambda_2 > 0$

P3. Consider the system

$$\dot{\theta}_i(t) = \omega_i + \sum_{j \in N(i)} sin(\theta_j(t) - \theta_i(t)), \text{ for } i = 1, 2, 3, \cdots, n$$

which resembles the agreement protocol with the linear term $x_j - x_i$ replaced by the nonlinear term $sin(\theta_j(t) - \theta_i(t))$. For $\omega_i = 0$, simulate (4.35) for $n = 5$ and various connected graphs on five nodes. Do the trajectories of (4.35) always converge for any initialization? How about for $\omega_i \neq 0$? (This is a "simulation-inspired question" so it is okay to conjecture!)

P4. Provide an example for an agreement protocol on a digraph that always converges to the agreement subspace (from arbitrary initializa- tion), yet does not admit a quadratic Lyapunov function of the form 12 xT x, that testifies to its asymptotic stability with respect to the agreement sub- space.

P5. Let $\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n$ be the ordered eigenvalues of the graph Laplacian associated with an undirected graph. We have seen that the second eigenvalue λ_2 is important both as a measure of the robustness in the graph, and as a measure of how fast the protocol converges. Given that our job is to build up a communication network by incrementally adding new edges (communication channels) between nodes, it makes sense to try and make λ_2 as large as possible.
Write a program that iteratively adds edges to a graph (starting with a connected graph) in such a way that at each step, the edge (not necessarily unique) is added that maximizes λ_2 of the graph Laplacian associated with the new graph. In other words, implement the following algorithm:
Step 0: Given G0 a spanning tree on n nodes. Set k=0
Step 1: Add a single edge to produce Gnew from Gk such that lambda2(Gnew) is maximized. Set k=k+1, Gk=Gnew
Repeat Step 1 until Gk=Kn for n = 10, 20, and 50. Did anything surprising happen?