ME-GY 7943/ECE-GY9273
Network Robotic Systems, Cooperative Control and Swarming

Exercise series 2

Please typeset your answers (e.g. using LaTeX). For all questions, justify clearly your answers. Include plots where requested, either in a Jupyter Notebook or in the typesetted answers. For questions requesting a software implementation, please provide your code in a python file or in a Jupyter Notebook such that it can be run directly. Include comments explaining how the functions work and how the code should be run if necessary. Any piece of code that does not run out of the box or does not contain instructions to execute it will be considered invalid.

Exercise 1

In python

a) Write a function *get_laplacian* that takes as input a list of edges (i.e. each element of the list is itself a list that contains the tail and head vertices defining the edge), the number of vertices and a Boolean flag on whether the graph is directed or not and that returns the graph Laplacian as a numpy array. Vertices are numbered starting at 0. For example:

```
# list of edges for a graph containing 3 nodes: 0, 1 and 2
E = [[0,1],[1,2],[2,0]]
n_vertices = 3

# construct an undirected graph
laplacian = get_laplacian(E, n_vertices, False)
print(laplacian)

# construct a directed graph
laplacian = get_laplacian(E, n_vertices, True)
print(laplacian)
```

would print first

$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \tag{1}$$

for the first output which is a the Laplacian for the undirected graph and

$$\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \tag{2}$$

which is the Laplacian for a directed graph given the set of edges.

b) Test this function to compute the Laplacian of the graphs computed in Exercises 1 and 2.

c) Using this function compute the two smallest and the two largest eigenvalues of the cycle graph C_5, C_{15} and C_{199}. You can use sorting functions[1] and linear algebra functions to compute eigenvalues[2]. You are also advised to automatically create the set of edges E (e.g. using a for loop).

[1]https://docs.scipy.org/doc/numpy/reference/generated/numpy.sort.html
[2]https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eigvals.html#numpy.linalg.eigvals

Exercise 2

Using the *get_laplacian* function computed previously, we will simulate, in matrix form, the consensus protocol for the temperature measurement example shown in class and whose communication graph is shown in Figure 1

a) Write a function

```python
def simulate_consensus(x_0, T, L, dt=0.001):
```

that takes as input a vector of size n of conditions x_0, a desired integration time T, a graph Laplacian L and an optional integration time dt (which is 0.001 by default) and integrates the consensus protocol as the differential equations $\dot{x} = -Lx$ from $t = 0$ to $t = T$ and returns a vector t containing the time from 0 to T discretized every dt and a matrix (numpy.array) x of size $n \times \frac{T}{dt}$ that contains all n robot states at each instant of time (i.e. $x[i, j]$ contains the state of robot i at time $t[j]$). Use the Euler integration scheme seen in class to do the integration.

b) Simulate the consensus protocol for the initial temperature measurements

$$x_0 = [10, 20, 12, 5, 30, 12, 15, 16, 25] \tag{3}$$

Plot the state of every robot as a function of time in a single graph. How much time does it take until the states reach consensus? (assume that consensus is reached when the states are within a distance of 0.01 of each other). How does the result compare to the theoretical prediction?

c) How will the convergence change if we remove the edges between 2 and 6 and between 2 and 4? Verify your prediction experimentally.

d) How will the convergence change if a complete graph is used instead? Compare your theoretical prediction with a simulation[3].

e) Same question as d) for a cycle graph.

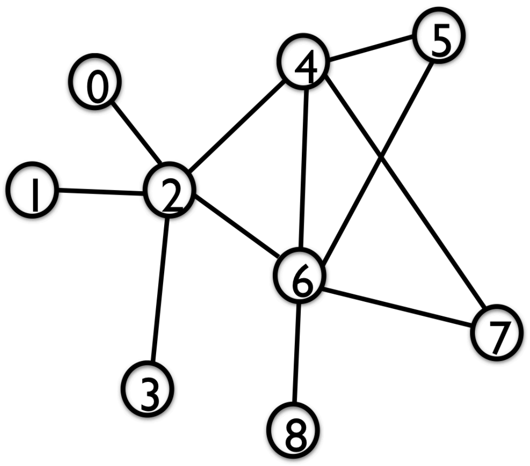


Figure 1: Communication network for the temperature measuring drones

Exercise 3

Consider a directed graph with graph Laplacian $\mathbf{L}$ and assume that the left eigenvalues of $\mathbf{L}$ are ordered as $\lambda_1 \leq \lambda_2 \leq \cdots$ with respective left eigenvectors $\mathbf{q}_1^T, \mathbf{q}_2^T, \cdots$.

a) Under which conditions is $\lambda_2 > 0$? Explain why.

[3]You may want to write a function that creates a list of edges for a fully connected graph instead of writing the list by hand

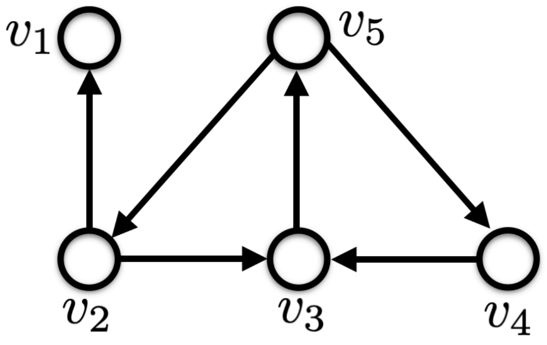
b) Prove that for any graph Laplacian $\mathbf{L}$, the quantity $\mathbf{q}_1^T\mathbf{x}$ is a constant of motion under the consensus dynamics $\dot{\mathbf{x}} = -\mathbf{L}\mathbf{x}$.

c) Consider the consensus algorithm applied for the graph below. To which value will the states converge for initial conditions $x_1 = 10$, $x_2 = 5$, $x_3 = 1$, $x_4 = -5$, $x_5 = -10$?

d) Add or remove edges to this graph to ensure that the consensus algorithm converges to the average of the initial conditions (show the graph). What is then the value of $\mathbf{q}_1$? (Normalize the vector such that $\mathbf{q}_1^T\mathbf{1} = 1$).

e) Add or remove edges to this graph to ensure that the consensus algorithm converges to the initial value of v_4 (show the graph). What is then the value of $\mathbf{q}_1$? (Normalize the vector such that $\mathbf{q}_1^T\mathbf{1} = 1$).

f) Simulate the consensus protocol for each of the previous 3 scenarios to verify your predictions. Show for each scenario the time evolution of the states and plot a black line showing the predicted consensus.


v_1
v_5
v_2
v_3
v_4