Homework 8

Name:

Student ID:

- Reasoning and work must be shown to gain partial/full credit
- Please include the cover-page on your homework PDF with your name and student ID. Failure of doing so is considered bad citizenship.

1. (1–4 points) **Lecture 13 Question**: You want to simulate the following network dynamics: You want to simulate the SIR model over the network hospital_edge.txt with different values of $\beta = (0.2, 0.5, 0.9)$ and $\gamma = 0.1$, where d_i is the degree of node i.

   (a) (50% points) Use the finite difference method to calculate the evolution of the probability of infection where dt is the unit of time (for simplicity choose $dt = 1$):

   $$\begin{aligned} \dot{x}_i(t) = \frac{x_i(t+1) - x_i(t)}{dt} &= \sum_{j \in \mathcal{N}_i} \frac{\beta}{d_i}(1 - x_i(t))x_j(t) - \gamma x_i(t) \\ &= \frac{\beta}{d_i}(1 - x_i(t)) \sum_{j \in \mathcal{N}_i} x_j(t) - \gamma x_i(t) \end{aligned} \tag{1}$$

   setting the initial state $\boldsymbol{x}(0) = \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is a coordinate vector for node i. Show the evolution of all nodes the states over time (similar to the plots shown in Lecture 12, slide 16, at the bottom)

   (b) (50% points) Consider now the SI model, which essentially is the same as the SIR but with $\gamma = 0$ (nobody is removed). We want to model the actual random infections rather than evaluating their probability. We can simulate the state $y_i(t) \in \{0, 1\}$ of each individual as follows.

   - We start with one infected node $y_1(0) = 1$ and all the rest $y_i(0) = 0$ $i \in \mathcal{V}/i$
   - At each t each node i selects uniformly at random a neighbor $j \in \mathcal{N}_i$ with probability $\beta/|\mathcal{N}_i|$ or, does not meet anyone with probability $(1 - \beta)$.
   - Let $m_{ij}(t)$ be the *meeting indicator function* that is 1 if node i meets node j at time t and zero else. Then:

   $$y_i(t+1) = y_i(t) + \sum_{j \in \mathcal{N}_i} m_{ij}(t)(1 - y_i(t))y_j(t), \qquad i \in \mathcal{V} \tag{2}$$

   Simulate several realizations of $y_i(t)$ with $\beta = 0.2$ and show that on average across 1000 experiments you obtain the same result as $x_i(t)$ for the SI model (the same as equation (1) with $\beta = 0.2$ and $\gamma = 0$).

2. (2 points) **Hengselmann-Krause bounded confidence model**:

   (a) (100% points) For a given network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the Hengselmann-Krause bounded confidence interval update rule is given by:

   $$x_i(t+1) = \frac{\sum_{j \in \mathcal{V}} W_{ij} \cdot u_\tau(d_{ij}(x_i(t), x_j(t))) \cdot x_j(t)}{\sum_{j \in \mathcal{V}} W_{ij} \cdot u(\tau - d_{ij}(x_i(t), x_j(t)))}, \qquad \forall i \in \mathcal{V},$$

   where

   - $x_i(t)$ is the state of node i at time t,
   - $d_{ij}(x_i, x_j) = |x_i - x_j|$ is the distance between two states,
   - $u_\tau(\cdot)$ is an indicator function that decides if two nodes are in each others' confidence, and it is given by:

   $$u_\tau(y) = \begin{cases} 1, & y \leq \tau \\ 0, & \text{else} \end{cases},$$

- $\boldsymbol{W} = (1-\alpha)\boldsymbol{I} + \alpha \boldsymbol{L}^{\text{rw}}$ is the mixing matrix with $\alpha = \frac{0.5}{\max_i(d_i)}$, where $\max_i(d_i)$ is the maximum degree.

Draw the initial opinion of agents from i.i.d uniform distribution between 0 and 1 and simulate the Hengselmann-Krause bounded confidence model on the network given in `rt-retweet.mtx` by considering two thresholds $\tau = 0.1$ and $\tau = 0.5$. Plot the evolution of the opinion of the agents.

3. (3 points) **Lecture 15 problem**:
   (a) (50% points) Simulate electric grid power flows for IEEE case 30. This test circuit has $N = 30$ buses indexed from 0 to 29. Please find its admission matrix, $\boldsymbol{Y} \in \mathbb{C}^{N\times N}$, susceptance matrix, $\boldsymbol{B} \in \mathbb{R}^{N\times N}$, and voltage phasors, $\boldsymbol{v} \in \mathbb{C}^N$, in the attached file ("ieee30.mat")[1].
      i. (25%) Given the voltage phasors, $\boldsymbol{v}$, calculate the apparent power generations of all the buses, $\boldsymbol{s} \in \mathbb{C}^N$.
      ii. (25%) Consider the DC Power Flow equations (in Slide 37).
         - Calculate the active (real) power injections, $\boldsymbol{p} \in \mathbb{R}^N$, from the apparent powers from part i.
         - Use the active power injections to compute all the voltage angles, $\boldsymbol{\theta} \in \mathbb{R}^N$, of all the buses.
           While performing load flow studies, one needs to use a slack bus to balance the active and reactive powers in the system. Thus, assume bus 0 as the slack bus and fix its voltage phase angle at 0 rad. Since the slack bus is used to balance the system, its active power generation is a variable that needs to be calculated as well. Therefore solve the following linear system:

$$\boldsymbol{p} = \boldsymbol{B}\boldsymbol{\theta},$$

           where p_0 and θ_i, for all $i \in \{1, \ldots, N-1\}$, are the unknowns.
   (b) (50% points) **Optimal Power Flow**: Import the case 30 bus system using the following code excerpt:

```
!pip install pypower
from pypower.case30 import case30

ppc = case30()
# \mathcal{N}
buses = ppc["bus"][:,0] - 1 # -1 because python is 0-indexed
# \mathcal{G}
generators = ppc["gen"][:,0] - 1 # -1 because python is 0-indexed
non_generators = list(set(buses) - set(generators))
demands = ppc["bus"][:, [3]]
g_low = ppc["gen"][:, [9]] # returns only the limits for the generators
                                    indexed by the elements in
                                    generators.
g_high = ppc["gen"][:, [8]] # returns only the limits for the generators
                                    indexed by the elements in
                                    generators.
```

   where $\mathcal{N}$ is the set of all buses with cardinality N and $\mathcal{G}$ is the set of all generators with cardinality G.

[1] Refer to Slides 34 for the definition.

i. (25%) Compute the AC Optimal Power Flow using `pypower.api.runopf(ppc)` and report the results.

ii. (25%) Optimal Power Flow is in general a non-convex optimization problem. But there are a lot of relaxation methods proposed in literature [1]. One of the easiest version is the DC power flow relaxations, where the the operator tries to minimize the total power generated such that the net power injected by a bus is equal to the difference of the power generated and the demand at that bus while also satisfying the generation capacity constraints. This may be written as a constrained optimization problem as follows:

$$\min_{\boldsymbol{g},\boldsymbol{\vartheta}} \; f_{cost} = \sum_{i \in \mathcal{G}} g_i \tag{3}$$

$$\boldsymbol{g} - \boldsymbol{d} = \boldsymbol{B}\boldsymbol{\vartheta} \tag{4}$$

$$\underline{\boldsymbol{g}} \leq \boldsymbol{g} \leq \overline{\boldsymbol{g}} \tag{5}$$

$$g_i = 0, \quad \forall i \in \mathcal{N} \setminus \mathcal{G}, \tag{6}$$

where

- $\boldsymbol{g} = [g_0, \cdots, g_{N-1}]^\top$ is the vector of active power generations,
- $\boldsymbol{d} = [d_0, \cdots, d_{N-1}]^\top$ is the vector of demands,
- $\boldsymbol{B}$ is the susceptance matrix, and
- $\boldsymbol{\vartheta} = [\vartheta_0, \cdots, \vartheta_{N-1}]^\top$ is the vector of voltage phase angles.

Please utilize the convex programming solver cvxpy[2] to solve for the optimal $\boldsymbol{g}$ and $\boldsymbol{\vartheta}$.

References

[1] Steven H Low. Convex relaxation of optimal power flow—part i: Formulations and equivalence. *IEEE Transactions on Control of Network Systems*, 1(1):15–27, 2014.

[2] `https://www.cvxpy.org/examples/basic/linear_program.html`