Homework 1

EE 263 Stanford University

Summer 2017

1. **A simple power control algorithm for a wireless network.** First some background. We consider a network of n transmitter/receiver pairs. Transmitter i transmits at power level p_i (which is positive). The path gain from transmitter j to receiver i is G_{ij} (which are all nonnegative, and G_{ii} are positive). The signal power at receiver i is given by $s_i = G_{ii}p_i$. The noise plus interference power at receiver i is given by

$$q_i = \sigma^2 + \sum_{j \neq i} G_{ij} p_j$$

where $\sigma^2 > 0$ is the self-noise power of the receivers (assumed to be the same for all receivers). The *signal to interference plus noise ratio* (SINR) at receiver i is defined as $S_i = s_i/q_i$. For signal reception to occur, the SINR must exceed some threshold value γ (which is often in the range $3 - 10$). Various *power control algorithms* are used to adjust the powers p_i to ensure that $S_i \geq \gamma$ (so that each receiver can receive the signal transmitted by its associated transmitter). In this problem, we consider a simple power control update algorithm. The powers are all updated synchronously at a fixed time interval, denoted by $t = 0, 1, 2, \ldots$. Thus the quantities p, q, and S are discrete-time signals, so for example $p_3(5)$ denotes the transmit power of transmitter 3 at time epoch $t = 5$. What we'd like is

$$S_i(t) = s_i(t)/q_i(t) = \alpha\gamma,$$

where $\alpha > 1$ is an SINR safety margin (of, for example, one or two dB). Note that increasing $p_i(t)$ (power of the ith transmitter) increases S_i but decreases all other S_j. A very simple power update algorithm is given by

$$p_i(t+1) = p_i(t)(\alpha\gamma/S_i(t)). \qquad (1)$$

This scales the power at the next time step to be the power that would achieve $S_i = \alpha\gamma$, if the interference plus noise term were to stay the same. But unfortunately, changing the transmit powers also changes the interference powers, so it's not that simple! Finally, we get to the problem.

   a) Show that the power control algorithm (1) can be expressed as a linear dynamical system with constant input, *i.e.*, in the form

$$p(t+1) = Ap(t) + b,$$

   where $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ are constant. Describe A and b explicitly in terms of σ, γ, α and the components of G.

b) *matlab simulation.* Use matlab to simulate the power control algorithm (1), starting from various initial (positive) power levels. Use the problem data

$$G = \begin{bmatrix} 1 & .2 & .1 \\ .1 & 2 & .1 \\ .3 & .1 & 3 \end{bmatrix}, \qquad \gamma = 3, \qquad \alpha = 1.2, \qquad \sigma = 0.1.$$

Plot S_i and p as a function of t, and compare it to the target value $\alpha\gamma$. Repeat for $\gamma = 5$. Comment briefly on what you observe. *Comment:* You'll understand what you see later in the course.

2. State equations for a linear mechanical system. The equations of motion of a lumped mechanical system undergoing small motions can be expressed as

$$M\ddot{q} + D\dot{q} + Kq = f$$

where $q(t) \in \mathbb{R}^k$ is the vector of deflections, M, D, and K are the *mass*, *damping*, and *stiffness* matrices, respectively, and $f(t) \in \mathbb{R}^k$ is the vector of externally applied forces. Assuming M is invertible, write linear system equations for the mechanical system, with state

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix},$$

input $u = f$, and output $y = q$.

3. A mass subject to applied forces. Consider a unit mass subject to a time-varying force $f(t)$ for $0 \le t \le n$. Let the initial position and velocity of the mass both be zero. Suppose that the force has the form $f(t) = x_j$ for $j - 1 \le t < j$ and $j = 1, \ldots, n$. Let y_1 and y_2 denote, respectively, the position and velocity of the mass at time $t = n$.

a) Find the matrix $A \in \mathbb{R}^{2\times n}$ such that $y = Ax$.

b) For $n = 4$, find a sequence of input forces $x_1, \ldots, x_n$ that moves the mass to position 1 with velocity 0 at time n.

4. Counting paths in an undirected graph. Consider an undirected graph with n nodes, and no self loops (*i.e.*, all branches connect two different nodes). Let $A \in \mathbf{R}^{n\times n}$ be the *node adjacency matrix*, defined as

$$A_{ij} = \begin{cases} 1 & \text{if there is a branch from node } i \text{ to node } j \\ 0 & \text{if there is no branch from node } i \text{ to node } j \end{cases}$$

Note that $A = A^{\mathsf{T}}$, and $A_{ii} = 0$ since there are no self loops. We can intrepret A_{ij} (which is either zero or one) as the number of branches that connect node i to node j. Let $B = A^k$, where $k \in \mathbb{Z}$, $k \ge 1$. Give a simple interpretation of B_{ij} in terms of the original graph. (You might need to use the concept of a *path* of length m from node p to node q.)

5. **Counting sequences in a language or code.** We consider a language or code with an alphabet of n symbols $1, 2, \ldots, n$. A sentence is a finite sequence of symbols, $k_1, \ldots, k_m$ where $k_i \in \{1, \ldots, n\}$. A language or code consists of a set of sequences, which we will call the *allowable sequences*. A language is called *Markov* if the allowed sequences can be described by giving the allowable transitions between consecutive symbols. For each symbol we give a set of symbols which are allowed to follow the symbol. As a simple example, consider a Markov language with three symbols $1, 2, 3$. Symbol 1 can be followed by 1 or 3; symbol 2 must be followed by 3; and symbol 3 can be followed by 1 or 2. The sentence 1132313 is allowable (*i.e.*, in the language); the sentence 1132312 is not allowable (*i.e.*, not in the language). To describe the allowed symbol transitions we can define a matrix $A \in \mathbb{R}^{n \times n}$ by

$$A_{ij} = \begin{cases} 1 & \text{if symbol } i \text{ is allowed to follow symbol } j \\ 0 & \text{if symbol } i \text{ is not allowed to follow symbol } j \end{cases}.$$

a) Let $B = A^r$. Give an interpretation of B_{ij} in terms of the language.

b) Consider the Markov language with five symbols $1, 2, 3, 4, 5$, and the following transition rules:

- 1 must be followed by 2 or 3
- 2 must be followed by 2 or 5
- 3 must be followed by 1
- 4 must be followed by 4 or 2 or 5
- 5 must be followed by 1 or 3

Find the total number of allowed sentences of length 10. Compare this number to the simple code that consists of all sequences from the alphabet (*i.e.*, all symbol transitions are allowed). In addition to giving the answer, you must explain how you solve the problem. Do not hesitate to use matlab.

6. **Affine functions.** A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called *affine* if for any $x,\ y \in \mathbb{R}^n$ and any $\alpha,\ \beta \in \mathbb{R}$ with $\alpha + \beta = 1$, we have

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y).$$

(Without the restriction $\alpha + \beta = 1$, this would be the definition of linearity.)

a) Suppose that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Show that the function $f(x) = Ax + b$ is affine.

b) Now the converse: Show that any affine function f can be represented as $f(x) = Ax + b$, for some $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. (This representation is unique: for a given affine function f there is only one A and one b for which $f(x) = Ax + b$ for all x.)

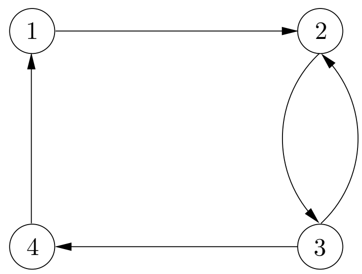
Hint. Show that the function $g(x) = f(x) - f(0)$ is linear.

You can think of an affine function as a linear function, plus an offset. In some contexts, affine functions are (mistakenly, or informally) called linear, even though in general they are not. (Example: $y = mx + b$ is described as 'linear' in US high schools.)

7. **Paths and cycles in a directed graph.** We consider a directed graph with n nodes. The graph is specified by its *node adjacency matrix* $A \in \mathbb{R}^{n \times n}$, defined as

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from node } j \text{ to node } i \\ 0 & \text{otherwise.} \end{cases}$$

Note that the edges are *oriented*, *i.e.*, $A_{34} = 1$ means there is an edge from node 4 to node 3. For simplicity we do not allow self-loops, *i.e.*, $A_{ii} = 0$ for all i, $1 \le i \le n$. A simple example illustrating this notation is shown below.



The node adjacency matrix for this example is

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

In this example, nodes 2 and 3 are connected in both directions, *i.e.*, there is an edge from 2 to 3 and also an edge from 3 to 2. A *path* of length $l > 0$ from node j to node i is a sequence $s_0 = j, s_1, \ldots, s_l = i$ of nodes, with $A_{s_{k+1}, s_k} = 1$ for $k = 0, 1, \ldots, l-1$. For example, in the graph shown above, $1, 2, 3, 2$ is a path of length 3. A *cycle* of length l is a path of length l, with the same starting and ending node, with no repeated nodes other than the endpoints. In other words, a cycle is a sequence of nodes of the form $s_0, s_1, \ldots, s_{l-1}, s_0$, with

$$A_{s_1, s_0} = 1, \quad A_{s_2, s_1} = 1, \quad \ldots \quad A_{s_0, s_{l-1}} = 1,$$

and

$$s_i \neq s_j \text{ for } i \neq j, \quad i, j = 0, \ldots, l-1.$$

For example, in the graph shown above, $1, 2, 3, 4, 1$ is a cycle of length 4. The rest of this problem concerns a specific graph, given in the file `directed_graph.m` on the course web site. For each of the following questions, you must give the answer explicitly (for example, enclosed in a box). You must also explain clearly how you arrived at your answer.

a) What is the length of a shortest cycle? (Shortest means minimum length.)

b) What is the length of a shortest path from node 13 to node 17? (If there are no paths from node 13 to node 17, you can give the answer as 'infinity'.)

c) What is the length of a shortest path from node 13 to node 17, that *does not* pass through node 3?

d) What is the length of a shortest path from node 13 to node 17, that *does* pass through node 9?

e) Among all paths of length 10 that start at node 5, find the most common ending node.

f) Among all paths of length 10 that end at node 8, find the most common starting node.

g) Among all paths of length 10, find the most common pair of starting and ending nodes. In other words, find i, j which maximize the number of paths of length 10 from i to j.