Homework 1 Solutions

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

$$p_i(t+1) = p_i(t)(\alpha\gamma/S_i(t)). \tag{1}$$

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.

Solution.

a) The power update rule for a single transmitter can be found by manipulating the definitions given in the problem.

$$p_i(t+1) = \frac{\alpha\gamma p_i(t)}{S_i(t)} = \frac{\alpha\gamma p_i(t) q_i(t)}{s_i(t)} = \frac{\alpha\gamma p_i(t)\left[\sigma^2 + \sum_{j\neq i} G_{ij} p_j(t)\right]}{G_{ii} p_i(t)}$$
$$= \frac{\alpha\gamma\left[\sigma^2 + \sum_{j\neq i} G_{ij} p_j(t)\right]}{G_{ii}}$$

In matrix form the equations look like this:

$$\underbrace{\begin{bmatrix} p_1(t+1) \\ p_2(t+1) \\ p_3(t+1) \\ \vdots \\ p_n(t+1) \end{bmatrix}}_{p(t+1)} = \underbrace{\begin{bmatrix} 0 & \frac{\alpha\gamma G_{12}}{G_{11}} & \frac{\alpha\gamma G_{13}}{G_{11}} & \cdots & \frac{\alpha\gamma G_{1n}}{G_{11}} \\ \frac{\alpha\gamma G_{21}}{G_{22}} & 0 & \frac{\alpha\gamma G_{23}}{G_{22}} & \cdots & \frac{\alpha\gamma G_{2n}}{G_{22}} \\ \frac{\alpha\gamma G_{31}}{G_{33}} & \frac{\alpha\gamma G_{32}}{G_{33}} & 0 & \cdots & \frac{\alpha\gamma G_{3n}}{G_{33}} \\ \vdots & & & \ddots & \\ \frac{\alpha\gamma G_{n1}}{G_{nn}} & \frac{\alpha\gamma G_{n2}}{G_{nn}} & \frac{\alpha\gamma G_{n3}}{G_{nn}} & \cdots & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \\ \vdots \\ p_n(t) \end{bmatrix}}_{p(t)} + \underbrace{\begin{bmatrix} \frac{\alpha\gamma\sigma^2}{G_{11}} \\ \frac{\alpha\gamma\sigma^2}{G_{22}} \\ \frac{\alpha\gamma\sigma^2}{G_{33}} \\ \vdots \\ \frac{\alpha\gamma\sigma^2}{G_{nn}} \end{bmatrix}}_{b}.$$

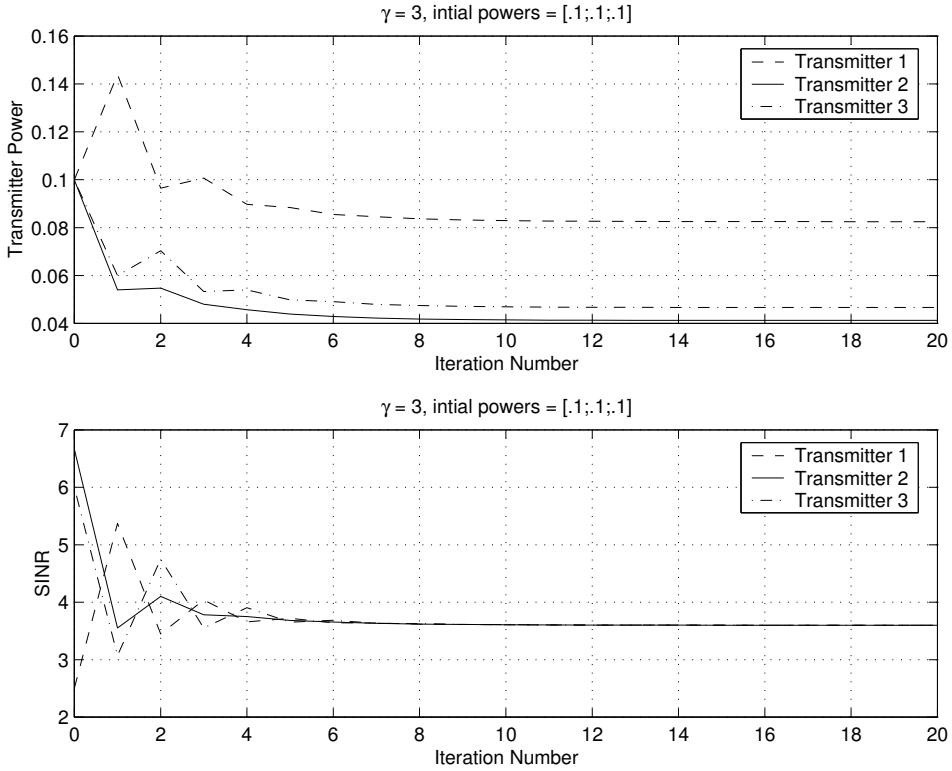
b) The following matlab code simulates the system for $\gamma = 3$ and an initial power of 0.1 for each transmitter.

```
clear all; close all;
G = [1 .2 .1; .1 2 .1; .3 .1 3];% Gain matrix
gamma = 3;                      % minimum SINR
alpha = 1.2;                    % safety margin
sigma = 0.01;                   % Noise power (same for all receivers)
A = zeros(3,3); for i = 1:3
for j = 1:3
if (i~=j)
A(i,j) = alpha*gamma*G(i,j)/G(i,i);
end
end
end
```

```
b = zeros(3,1); for i = 1:3
b(i) = alpha*gamma*sigma/G(i,i);
end
num_iterations = 20;
p_i = [.1;.1;.1]; % Initialized to p(0)
S = [G(1,1)*p_i(1)/(sigma+G(1,2)*p_i(2)+G(1,3)*p_i(3)); \
G(2,2)*p_i(2)/(sigma+G(2,1)*p_i(1)+G(2,3)*p_i(3)); \
G(3,3)*p_i(3)/(sigma+G(3,1)*p_i(1)+G(3,2)*p_i(2))];
p = p_i;      % matrix to store the powers versus time
for i = 1:num_iterations
p_i = A*p_i+b;
p = [p p_i]; % Find the new powers and save
SINR_current = [G(1,1)*p_i(1)/(sigma+G(1,2)*p_i(2)+G(1,3)*p_i(3)); \
G(2,2)*p_i(2)/(sigma+G(2,1)*p_i(1)+G(2,3)*p_i(3)); \
G(3,3)*p_i(3)/(sigma+G(3,1)*p_i(1)+G(3,2)*p_i(2))];
S = [S SINR_current];
end
figure(1); temp = 0:num_iterations; subplot(2,1,1);
plot(temp,p(1,:),'--', temp,p(2,:),'-',temp,p(3,:),'-.');
xlabel('Iteration Number'); ylabel('Transmitter Power');
title('\gamma = 3, intial powers = [.1;.1;.1]');
legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;
subplot(2,1,2); plot(temp,S(1,:),'--',
temp,S(2,:),'-',temp,S(3,:),'-.'); xlabel('Iteration Number');
ylabel('SINR'); title('\gamma = 3, intial powers = [.1;.1;.1]');
legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;
```

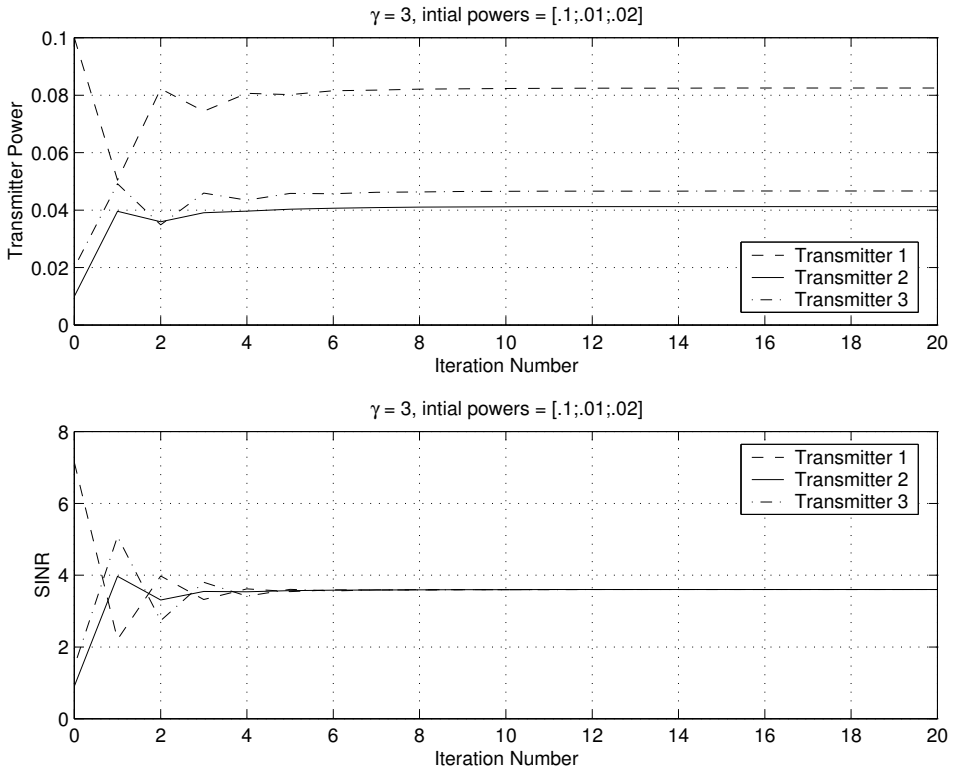
The figure below shows the SINR and transmitter power as a function of iteration num-

ber.



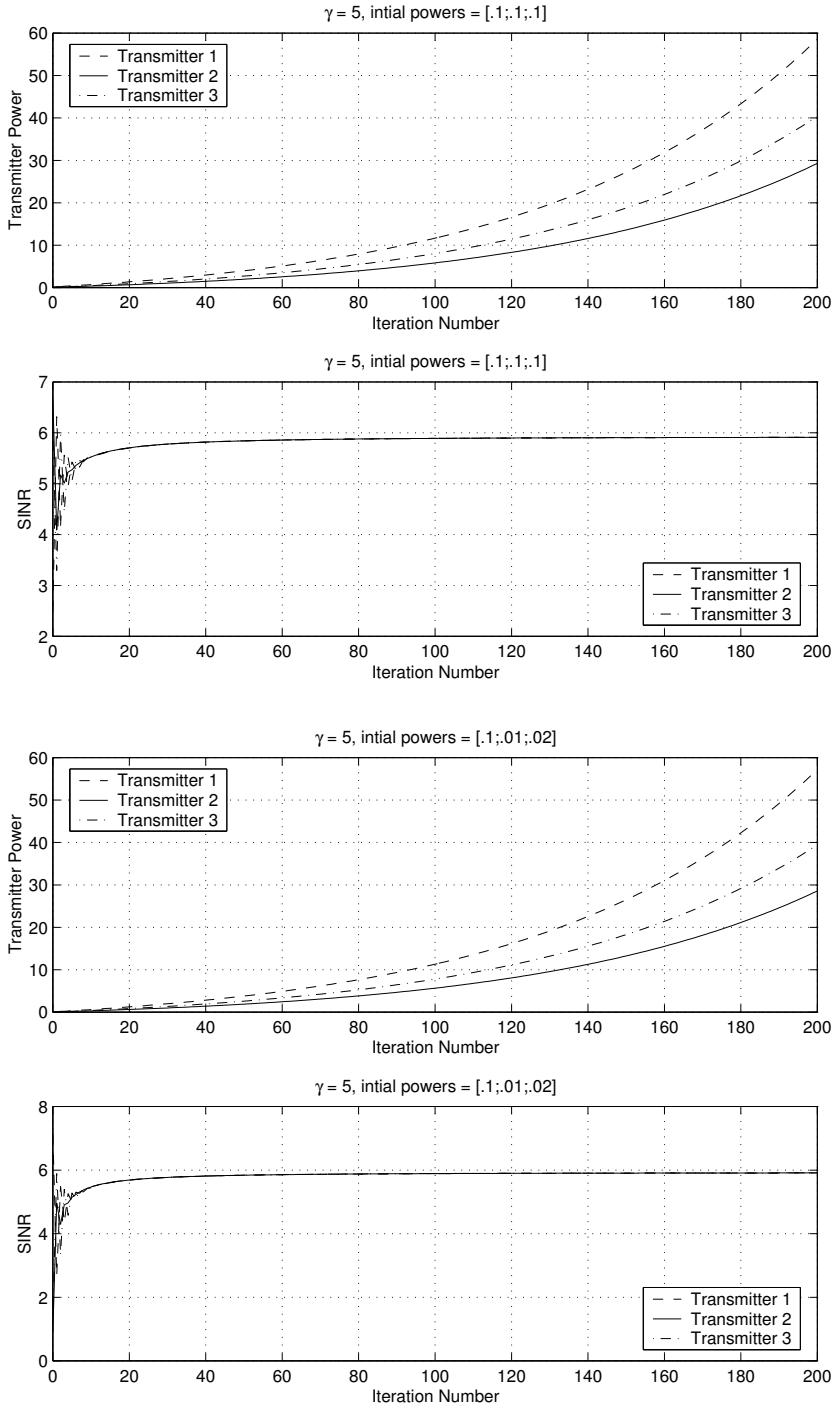
Similar matlab code can be used to try other initial transmitter powers. For example, the simulation shown below used initial transmitter powers of .1, .01, and .02 for the first, second, and third transmitter respectively. In both cases, the final transmitter powers approach .083, .041, and .047. The SINR appoaches $3.6 = \alpha\gamma$. The algorithm

appears to work.



Testing the system for $\gamma = 5$ and the same initial conditions (see graphs below) shows that the algorithm does not always succeed. For both initial conditions tried, the trans-

mitter powers grow exponentially. Also, the SINR approaches $5.92 < \alpha\gamma = 6$.



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 $y = q$.

Solution. We need to express the output q and the state derivative, $\dot{q}$ and $\ddot{q}$, as a linear function of the state variables q, $\dot{q}$ and the input f. In other words, we should find matrices A, B, C and D such that

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = A \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + Bf, \quad q = C \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + Df.$$

Matrices C and D are easy to find: simply, for the second equation to hold we should have

$$C = \begin{bmatrix} I & 0 \end{bmatrix}, \quad D = 0.$$

A and B are a bit harder to find. We will use the differential equation to express $\ddot{q}$ in terms of q, $\dot{q}$ and f. From the given dynamics equation $M\ddot{q} + D\dot{q} + Kq = f$, and assuming M is invertible, we get

$$\ddot{q} = -M^{-1}Kq - M^{-1}D\dot{q} + M^{-1}f,$$

which expresses $\ddot{q}$ in terms of q, $\dot{q}$, and f. Now we can write the linear dynamical system equations for the system. In block matrix notation we have

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} u, \quad y = \begin{bmatrix} I & 0 \end{bmatrix}\begin{bmatrix} q \\ \dot{q} \end{bmatrix},$$

so the matrices in linear dynamical system description are:

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}, \quad C = \begin{bmatrix} I & 0 \end{bmatrix}, \quad D = 0.$$

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

Solution. Let $p(t)$ and $v(t)$ denote, respectively, the position and velocity of the mass at time t.

a) The velocity is the integral of the applied force:

$$\begin{aligned}
v(t) &= v(0) + \int_0^t f(\tau)\,d\tau \\
&= v(0) + \sum_{j=1}^{\lfloor t \rfloor} \int_{j-1}^{j} f(\tau)\,d\tau + \int_{\lfloor t \rfloor}^{t} f(\tau)\,d\tau \\
&= v(0) + \sum_{j=1}^{\lfloor t \rfloor} \int_{j-1}^{j} x_j\,d\tau + \int_{\lfloor t \rfloor}^{t} x_{\lfloor t \rfloor + 1}\,d\tau \\
&= v(0) + \sum_{j=1}^{\lfloor t \rfloor} (\tau x_j)|_{\tau=j-1}^{\tau=j} + (\tau x_{\lfloor t \rfloor+1})|_{\tau=\lfloor t \rfloor}^{\tau=t} \\
&= v(0) + \sum_{j=1}^{\lfloor t \rfloor} x_j + (t - \lfloor t \rfloor) x_{\lfloor t \rfloor + 1}.
\end{aligned}$$

In particular, because the mass is initially at rest (that is, $v(0) = 0$), the final velocity is

$$y_2 = v(n) = \sum_{j=1}^{n} x_j.$$

Similarly, the position is the integral of the velocity:

$$\begin{aligned}
p(t) &= p(0) + \int_0^t v(\tau)\,d\tau \\
&= p(0) + \int_0^t (v(0) + (v(\tau) - v(0)))\,d\tau \\
&= p(0) + v(0)t + \int_0^t (v(\tau) - v(0))\,d\tau \\
&= p(0) + v(0)t + \sum_{j=1}^{\lfloor \tau \rfloor} \int_{j-1}^{j} (v(\tau) - v(0))\,d\tau + \int_{\lfloor t \rfloor}^{t} (v(\tau) - v(0))\,d\tau \\
&= p(0) + v(0)t + \sum_{j=1}^{\lfloor t \rfloor} \int_{j-1}^{j} \left(\sum_{k=1}^{\lfloor \tau \rfloor} x_k + (\tau - \lfloor \tau \rfloor) x_{\lfloor \tau \rfloor + 1} \right) d\tau \\
&\quad + \int_{\lfloor t \rfloor}^{t} \left(\sum_{k=1}^{\lfloor \tau \rfloor} x_k + (\tau - \lfloor \tau \rfloor) x_{\lfloor \tau \rfloor} + 1 \right) d\tau \\
&= p(0) + v(0)t + \sum_{j=1}^{\lfloor t \rfloor} \int_{j-1}^{j} \left(\sum_{k=1}^{j-1} x_k + (\tau - (j-1)) x_j \right) d\tau
\end{aligned}$$

$$
\begin{aligned}
&\quad + \int_{\lfloor t \rfloor}^{t} \left(\sum_{k=1}^{\lfloor t \rfloor} x_k + (\tau - \lfloor t \rfloor) x_{\lfloor t \rfloor + 1} \right) d\tau \\
&= p(0) + v(0)t + \sum_{k=1}^{\lfloor t \rfloor} \left(\sum_{k=1}^{j-1} \tau x_k + \frac{1}{2}(\tau - (j-1))^2 x_j \right) \Bigg|_{\tau=j-1}^{\tau=j} \\
&\quad + \left(\sum_{k=1}^{\lfloor t \rfloor} \tau x_k + \frac{1}{2}(\tau - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1} \right) \Bigg|_{\tau=\lfloor t \rfloor}^{\tau=t} \\
&= p(0) + v(0)t + \sum_{j=1}^{\lfloor t \rfloor} \left(\sum_{k=1}^{j-1} x_k + \frac{1}{2} x_j \right) + \left(\sum_{k=1}^{\lfloor t \rfloor} x_k + \frac{1}{2}(t - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1} \right) \\
&= p(0) + v(0)t + \sum_{j=1}^{\lfloor t \rfloor} \sum_{k=1}^{j-1} x_k + \sum_{j=1}^{\lfloor t \rfloor} \frac{1}{2} x_j + \sum_{k=1}^{\lfloor t \rfloor} x_k + \frac{1}{2}(t - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1} \\
&= p(0) + v(0)t + \sum_{k=1}^{\lfloor t \rfloor} \sum_{j=k+1}^{\lfloor t \rfloor} x_k + \sum_{k=1}^{\lfloor t \rfloor} \frac{1}{2} x_k + \sum_{k=1}^{\lfloor t \rfloor} x_k + \frac{1}{2}(t - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1} \\
&= p(0) + v(0)t + \sum_{k=1}^{\lfloor t \rfloor} (\lfloor t \rfloor - k) x_k + \sum_{k=1}^{\lfloor t \rfloor} \frac{1}{2} x_k + \sum_{k=1}^{\lfloor t \rfloor} x_k + \frac{1}{2}(t - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1} \\
&= p(0) + v(0)t + \sum_{k=1}^{\lfloor t \rfloor} \left((\lfloor t \rfloor - k) + \frac{1}{2} + (t - \lfloor t \rfloor) \right) x_k + \frac{1}{2}(t - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1} \\
&= p(0) + v(0)t + \sum_{k=1}^{\lfloor t \rfloor} \left(t - k + \frac{1}{2} \right) x_k + \frac{1}{2}(t - \lfloor t \rfloor)^2 x_{\lfloor t \rfloor + 1}.
\end{aligned}
$$

In particular, because the mass is initially at rest at the origin (that is, $p(0) = 0$ and $v(0) = 0$), the final position is

$$
y_1 = p(n) = \sum_{j=1}^{n} \left(n - k + \tfrac{1}{2} \right) x_j.
$$

Thus, we obtain the following system of linear equations:

$$
\begin{aligned}
y_1 &= \sum_{j=1}^{n} (n - j + \tfrac{1}{2}) x_j, \\
y_2 &= \sum_{j=1}^{n} x_j.
\end{aligned}
$$

Since A_{ij} gives the coefficient of x_j in our expression for y_i, we have that

$$
A_{1j} = n - j + \tfrac{1}{2} \qquad \text{and} \qquad A_{2j} = 1, \qquad j = 1, \ldots, n.
$$

More concretely, we have that

$$A = \begin{bmatrix} n - \frac{1}{2} & n - \frac{3}{2} & \cdots & \frac{3}{2} & \frac{1}{2} \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}.$$

b) We want to solve the following system of linear equations:

$$\begin{bmatrix} \frac{7}{2} & \frac{5}{2} & \frac{3}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

This system is underdetermined, and has infinitely many solutions. Suppose we choose $x_2 = x_3 = 0$. Then, we are left with the system

$$\begin{bmatrix} \frac{7}{2} & \frac{1}{2} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The second equation implies that $x_4 = -x_1$. Then, the first equation becomes

$$\tfrac{7}{2}x_1 + \tfrac{1}{2}x_4 = \tfrac{7}{2}x_1 - \tfrac{1}{2}x_1 = 3x_1 = 1.$$

Solving this equation, we find that $x_1 = \frac{1}{3}$. Substituting this value into our expression for x_4 gives $x_4 = -x_1 = -\frac{1}{3}$. Thus, one sequence of input forces that moves the mass to position 1 with velocity 0 at time n is

$$x = \begin{bmatrix} \frac{1}{3} \\ 0 \\ 0 \\ -\frac{1}{3} \end{bmatrix}.$$

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 $k \in \mathbb{Z}$, $k \geq 1$. 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.)

Solution. First consider $B = A$, *i.e.*, $(k = 1)$. Obviously, the interpretation of B_{ij} is the number of branches that connect node i to node j (either 0 or 1). In other words, B_{ij} is equal to the number of paths of length 1 that connect node i to node j. Now consider the case $k = 2$ so $B = A^2$ and

$$B_{ij} = \sum_m A_{im} A_{mj}.$$

Clearly, B_{ij} becomes the number of paths of length 2 from node i to node j. $A_{im}A_{mj}$ is nonzero only when both A_{im} and A_{mj} are nonzero so that there exists a path of length 2 from node i to node j via node m. The summation is over *all* nodes m and $A_{im}A_{mj}$ is either 0 or 1, so in fact, B_{ij} sums up to the number of paths of length 2 from node i to node j. For $k = 3$, $B = A^3$ and therefore

$$B_{ij} = \sum_{m_1} \sum_{m_2} A_{im_1} A_{m_1 m_2} A_{m_2 j}.$$

For similar reasons, B_{ij} now becomes the number of paths of length 3 from node i to node j. The summation is over all intermediate nodes m_1 and m_2, and $A_{im_1}A_{m_1m_2}A_{m_2j} = 1$ means that there is a path of length 3 from node i to node j via nodes m_1 and m_2. In general, for $B = A^k$, B_{ij} has the interpretation of "the number of paths of length k from node i to j" because

$$B_{ij} = \sum_{m_1} \sum_{m_2} \cdots \sum_{m_{k-1}} A_{im_1} A_{m_1 m_2} \cdots A_{m_{k-1} j}$$

and $A_{im_1}A_{m_1m_2} \cdots A_{m_{k-1}j} = 1$ means that there is a path of length k from node i to node j via nodes $m_1, m_2, \ldots, m_{k-1}$.

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.

Solution.

a) If $B = A^k$, then B_{ij} is the number of sequences of length $k+1$ that start with symbol j and end with symbol i.

Here is a formal proof. Let $S_L(i,j)$ denote the set of sentences of length L that start with symbol j and end with symbol i:

$$S_L(i,j) = \{(k_1 = j, k_2, \ldots, k_{L-1}, k_L = i) \in \mathbb{N}_n^L \mid A_{k_\ell k_{\ell+1}} = 1 \text{ for all } \ell = 1, \ldots, L-1\}.$$

We claim that

$$(A^p)_{ij} = |S_{p+1}(i,j)|$$

for all $i, j \in \mathbb{N}_n$ and $p \in \mathbb{N}$. First, we introduce some notation. Given a finite sequence of symbols $(k_1, \ldots, k_L)$ and a symbol k_{L+1}, we define

$$(k_1, \ldots, k_L) + k_{L+1} = (k_1, \ldots, k_L, k_{L+1}).$$

(Thus, $+$ denotes the operation of appending a symbol to a finite sequence of symbols.) Similarly, given a set S of finite sequences of symbols, we define

$$S + j = \{s + j \mid s \in S\}.$$

Finally, we define

$$\phi(i) = \{j \in \mathbb{N}_n \mid \text{symbol } i \text{ is allowed to follow symbol } j\} = \{j \in \mathbb{N}_n \mid A_{ij} = 1\}.$$

First, we prove the claim when $p = 1$:

$$(A^1)_{ij} = |S_2(i,j)|.$$

Note that $|S_2(i,j)| = 1$ if (j,i) is a sentence in the language (that is, symbol i is allowed to follow symbol j; or, equivalently, $A_{ij} = 1$), and $|S_2(i,j)| = 0$ otherwise (that is, symbol i is not allowed to follow symbol j; or, equivalently, $a_{ij} = 0$). In either case, we have that $|S_2(i,j)| = A_{ij} = (A^1)_{ij}$. Now suppose that $(A^p)_{ij} = |S_{p+1}(i,j)|$ for some $p \in \mathbb{N}$. We can partition $S_{p+2}(i,j)$ based on the penultimate symbol:

$$S_{p+2}(i,j) = \bigsqcup_{k \in \phi(i)} (S_{p+1}(k,j) + i).$$

Since the size of a disjoint union is equal to the sum of the sizes of the sets forming the union, we have that

$$|S_{p+2}(i,j)| = \sum_{k \in \phi(i)} |S_{p+1}(k,j) + i| = \sum_{k \in \phi(i)} |S_{p+1}(k,j)|.$$

Because $A_{ik} = 1$ for $k \in \phi(i)$, and $A_{ik} = 0$ for $k \notin \phi(i)$, we have that

$$|S_{p+2}(i,j)| = \sum_{k=1}^{n} A_{ik} |S_{p+1}(k,j)|.$$

Using the induction hypothesis, we have that $|S_{p+1}(k,j)| = (A^p)_{kj}$, and hence that

$$|S_{p+2}(i,j)| = \sum_{k=1}^{n} A_{ik} (A^p)_{kj} = (AA^p)_{ij} = (A^{p+1})_{ij}.$$

By induction, this proves that our interpretation of $(A^p)_{ij}$ is correct for all $p \in \mathbb{N}$.

b) For the given Markov language we can find the number of allowed sequences of length 10 by simply adding all the entries of the matrix A^9. Form the description of the rules we have

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

Using the matlab command `B = A^9` we find

$$B = A^9 = \begin{bmatrix} 41 & 49 & 24 & 113 & 37 \\ 55 & 65 & 31 & 150 & 49 \\ 42 & 49 & 23 & 113 & 37 \\ 0 & 0 & 0 & 1 & 0 \\ 31 & 37 & 18 & 86 & 28 \end{bmatrix}.$$

(Note that there is only one word that ends with 4 (*i.e.*, 4444444444, because 4 can only follow 4.) Adding all the elements of B, using for example the matlab command `sum(sum(B))`, we find that the total number of allowed sequences of length 10 is 1079. Finally, we can compare this number to the simple code that consists of all sequences from the alphabet. Of course there are $5^{10} = 9765625$ such sequences. Just to check our method, we can also compute this number the same way as above, by forming the matrix

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

(which means any symbol can follow any symbol), then find $B = A^9$, and adding all the entries of the matrix B. (Yes, you do get the same number as above ...)

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.)

Solution.

a) With $f(x) = Ax + b$, we have

$$\begin{aligned} f(\alpha x + \beta y) &= A(\alpha x + \beta y) + b \\ &= \alpha Ax + \alpha b + \beta Ay + (1 - \alpha)b \\ &= \alpha(Ax + b) + \beta(Ay + b) \\ &= \alpha f(x) + \beta f(y), \end{aligned}$$

and thus f is affine.

b) Assume that f is affine; we'll show that $g(x) = f(x) - f(0)$ is linear. First we show that $g(\alpha x) = \alpha g(x)$ for any $x \in \mathbb{R}^n$ and any $\alpha \in \mathbb{R}$.

$$\begin{aligned} g(\alpha x) &= f(\alpha x) - f(0) \\ &= f(\alpha x + (1 - \alpha)0) - f(0) \\ &= \alpha f(x) + (1 - \alpha)f(0) - f(0) \\ &= \alpha(f(x) - f(0)) \\ &= \alpha g(x), \end{aligned}$$

where the third line follows from affineness of f (since $\alpha + \beta = 1$ when $\beta = 1 - \alpha$). To establish linearity of g, we must also show that $g(x + y) = g(x) + g(y)$. We do this as follows.

$$\begin{aligned} g(x + y) &= f(x + y) - f(0) \\ &= f\left((1/2)(2x) + (1/2)(2y)\right) - f(0) \\ &= (1/2)f(2x) + (1/2)f(2y) - f(0) \\ &= (1/2)(f(2x) - f(0)) + (1/2)(f(2y) - f(0)) \\ &= (1/2)g(2x) + (1/2)g(2y) \\ &= g(x) + g(y). \end{aligned}$$

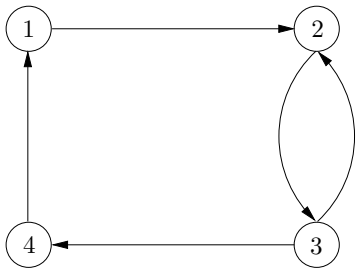
The third line is by affineness of f. The last line uses the result above, *i.e.*, $g(\alpha z) = \alpha g(z)$ for any α and z. So now we know that g is linear. It follows that there is an $A \in \mathbb{R}^{m \times n}$ for which $g(x) = Ax$ for any x. Thus we have $f(x) = g(x) + f(0) = Ax + f(0)$. With $b = f(0)$, we see that $f(x) = Ax + b$.

You might be interested in a way to find A and b directly from the affine function f. This is done as follows. First we set $b = f(0)$. Then we set $a_i = f(e_i) - b$, for $i = 1, \ldots, n$, where e_i is the ith unit vector. Then we have $A = [a_1 \cdots a_n]$. So to find A and b, you need to evaluate f a total of $n + 1$ times. Thereafter, we can *predict* what $f(x)$ will be, for *any* x, using the form $f(x) = Ax + b$.

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, $1 \leq i \leq n$. 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.

Solution.

a) Recall that $(A^k)_{ij}$ gives the number of paths of length k from node j to node i. Thus, $(A^k)_{ii}$ is the number of paths of length k from node i to itself. Now imagine increasing k from $k = 1$ to $k = 2$, $k = 3$, and so on. We find the smallest k for which $(A^k)_{ii} > 0$. This k is the length of the smallest path from i to itself. This path is in fact also a cycle, since it cannot repeat nodes. (If it repeated nodes, there would have been a shorter path from i to itself.) Now let's solve the problem. To find the length of a shortest cycle, find the smallest k such that $(A^k)_{ii} > 0$ for some i. Note $k \leq n$, because if a cycle exists then it is at most of length n, where n is the number of nodes in the graph.

```
clear all
directed_graph;
sz = size(A);
n = sz(1); % number of nodes
cycle_found = 0;
for k = 1:n
if max(diag(A^k)) > 0
cycle_found = 1;
break;
end
end
if cycle_found == 1
length = k
else
fprintf('Graph contains no cycle.');
end
>>smallest_cycle_length
length = 6
```

The smallest cycle is of length 6.

b) To find the length of a shortest path from node 13 to node 17, find the smallest k such that $(A^k)_{17,13} > 0$.

```
path_found = 0;
for k = 1:n
Ak = A^k;
if Ak(17,13) > 0
path_found = 1;
break;
end
end
if path_found == 1
length = k
else
fprintf('No path from 13 to 17.')
```

```
end
>> shortest_path_13to17
length = 4
```

The shortest path from node 13 to node 17 is of length 4.

c) To find the shortest path from node 13 to node 17, that does not pass though node 3, remove node 3 from the graph and then find the shortest path from node 13 to node 17. The new adjacency matrix B for the graph is obtained by removing the 3rd row and column of the matrix A. Then find the smallest k such that $(B^k)_{17,13} > 0$.

```
B = [A(1:2, 1:2) A(1:2, 4:20);
A(4:20,1:2) A(4:20, 4:20)];
path_found = 0;
for k = 1:n-1
Bk = B^k;
if Bk(16,12)>0
path_found = 1;
break;
end
end
if path_found == 1
length = k
else
fprintf('No path exists.')
end
>> shortest_path_13to17not3
length = 5
```

The shortest path from node 13 to node 17, that does not pass through node 3 is of length 5.

d) To find the smallest path from node 13 to node 17, that does pass through node 9, find the shortest path from node 13 to node 9 and the shortest path from node 9 to node 17.

```
path_found = 0;
for k = 1:n
Ak = A^k;
if Ak(9,13) > 0
path_found = 1;
break;
end
end
if path_found == 1
length13to9 = k
else
fprintf('No path exists')
```

```
end
if path_found == 1
path_found = 0;
for k = 1:n
Ak = A^k;
if Ak(17,9) > 0
path_found = 1;
break;
end
end
if path_found == 1
length9to17 = k
else
fprintf('No path exists')
end
end
if path_found == 1
length = length13to9+length9to17
end
>> shortest_path_13to17thru9
length13to9 = 6
length9to17 = 4
length = 10
```

The shortest path from node 13 to node 17, that does pass through node 9 is of length 10.

e) The matrix A^{10} gives the number of paths of length 10, *i.e.*, $(A^{10})_{ij}$ is the number of paths of length 10 that start at node j and end at node i. The 5th column of the matrix A^{10} gives the number of paths of length 10 that start at node 5 and end at nodes $1, 2, \ldots, 20$ respectively. The index of the maximum entry of this column gives the most common ending node for paths of length 10 starting at node 5.

```
A10  = A^10;
start5 = A10(1:20,5);
[number, mostcommonendnode] = max(start5);
mostcommonendnode
>>endnode
mostcommonendnode = 5
```

The most common ending node for paths of length 10 starting at node 5, is 5.

f) The 8th row of the matrix A^{10} gives the number of paths that end at node 8. The index of the maximum entry of this row gives the most common starting node for paths of length 10 ending at node 8.

```
end8 = A10(8,1:20);
```

```
[number, mostcommonstartnode] = max(end8);
mostcommonstartnode
>>startnode
mostcommonstartnode = 8
```

The most common starting node for paths of length 10 ending at node 8, is 8.

g) The most common source/destination pair for paths of length 10 is the index of the maximum entry of A^{10}, *i.e.*, if (i, j) is the most common source/destination pair then no other number in the matrix A^{10} is greater than $(A^{10})_{ji}$.

```
[max_row, dests] = max(A10);
[max_e, mostcommonsource] = max(max_row);
mostcommonsource
mostcommondest= dests(mostcommonsource)
>>sdpair
mostcommonsource = 8
mostcommondest = 17
```

The most common source/destination pair for paths of length 10 is $(8, 17)$.