EE263 homework 1 solutions

2.1 *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

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

where $\sigma > 0$ 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 $A \in \mathbf{R}^{n\times n}$ and $b \in \mathbf{R}^n$ 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

$$G = \begin{bmatrix} 1 & .2 & .1 \\ .1 & 2 & .1 \\ .3 & .1 & 3 \end{bmatrix}, \qquad \gamma = 3, \qquad \alpha = 1.2, \qquad \sigma = 0.01.$$

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 soon understand what you see.

Solution:

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

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

In matrix form the equations look like this:

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

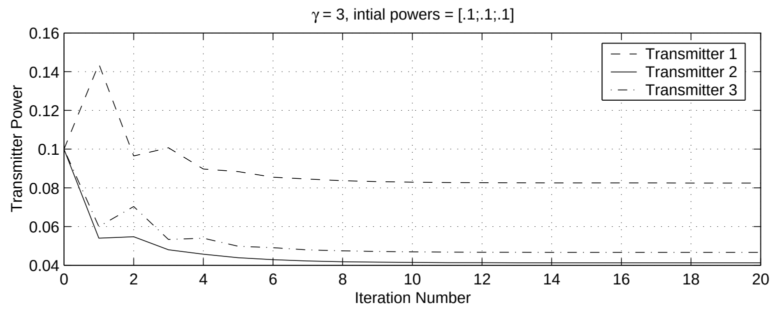
(b) The following matlab code simulates the system for $\gamma = 3$ and an initial power of 0.1 for each transmitter.

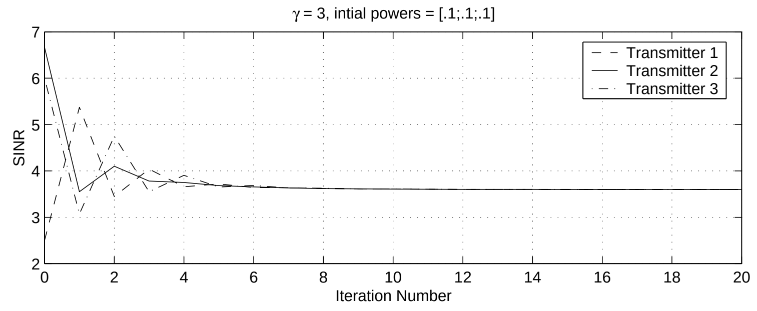
```
% NOTE:  The backslashes, \, that appear at the ends of some lines indicate
%        that the following line is a  continuation of the current one;  if
%        the code is enterred into matlab, they should be omitted and the
%        lines should be joined.
clear all; close all;

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)
```

```
% Form the A and b matrices
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

% Simulate
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))];
     % matrix  to store the SINR values versus time  (initialize to the
     % SINR for time 0)
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

% Plot the results
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]');
```

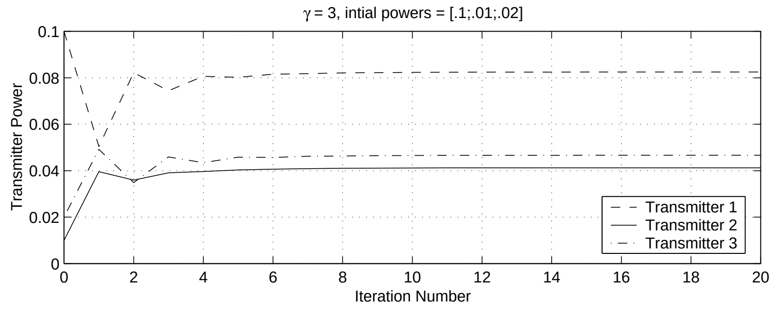
```
legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;
```

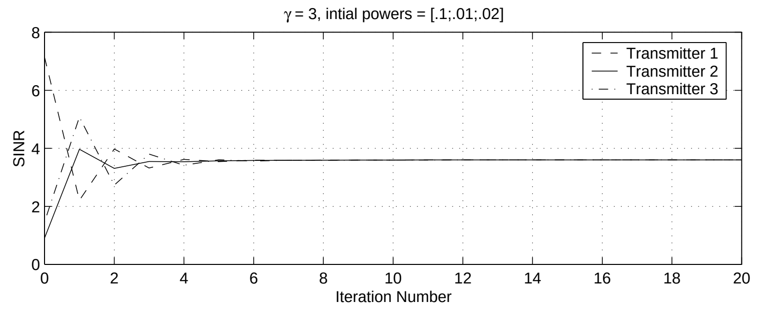
The figure below shows the SINR and transmitter power as a function of iteration number.



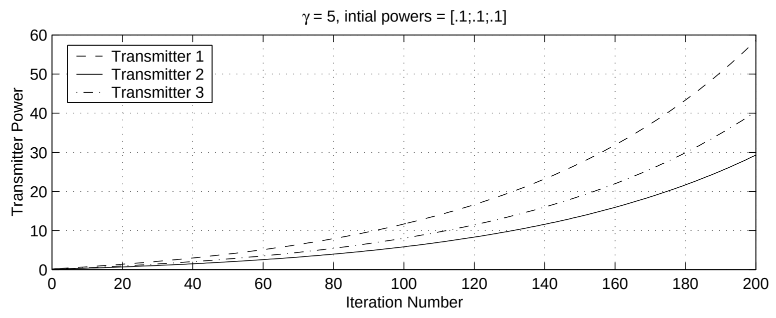


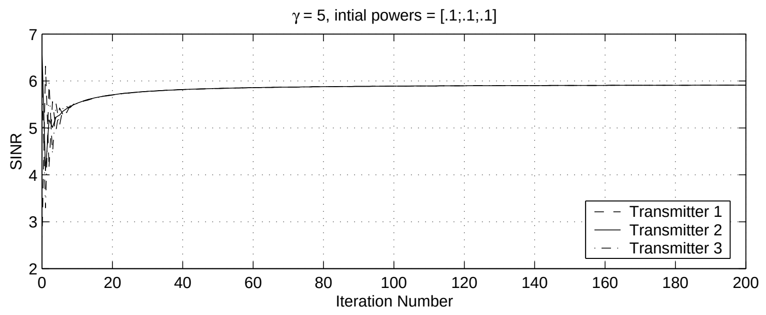
Similar matlab code can be used to try other initial transmitter powers. For example, the simulation shown below used initial transmitter powers of .1, .01, and .02 for the first, second, and third transmitter respectively. In both cases, the final transmitter powers approach .083, .041, and .047. The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

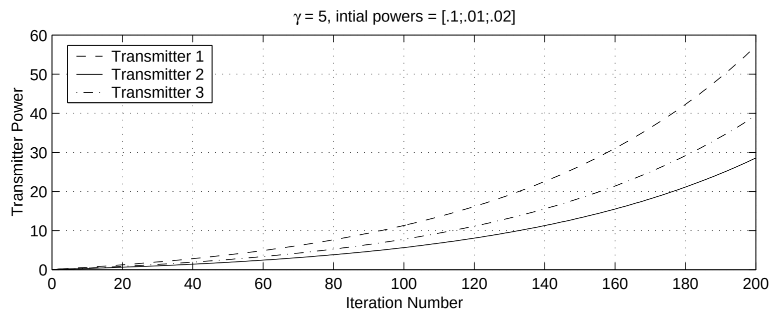


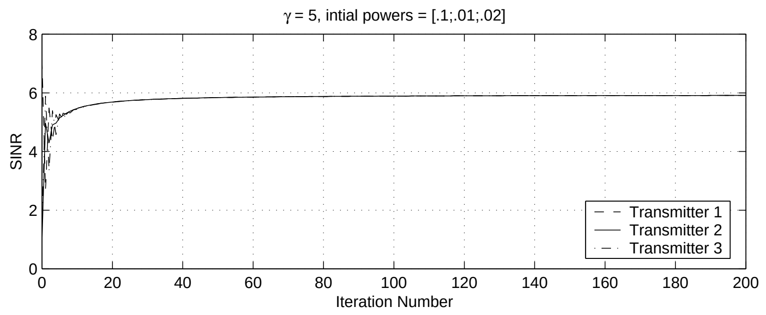


Testing the system for $\gamma = 5$ and the same initial conditions (see graphs below) shows that the algorithm does not always succeed. For both initial conditions tried, the transmitter powers grow exponentially. Also, the SINR approaches $5.92 < \alpha\gamma = 6$.









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

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

2.3 *Some standard time-series models.* A time series is just a discrete-time signal, *i.e.*, a function from $\mathbf{Z}_+$ into $\mathbf{R}$. We think of $u(k)$ as the value of the signal or quantity u at time (or *epoch*) k. The study of time series predates the extensive study of state-space linear systems, and is used in many fields (*e.g.*, econometrics).

Let u and y be two time series (input and output, respectively). The relation (or *time series model*)

$$y(k) = a_0u(k) + a_1u(k-1) + \cdots + a_ru(k-r)$$

is called a *moving average (MA) model,* since the output at time k is a weighted average of the previous r inputs, and the set of variables over which we average 'slides along' with time.

Another model is given by

$$y(k) = u(k) + b_1 y(k-1) + \cdots + b_p y(k-p).$$

This model is called an *autoregressive (AR) model*, since the current output is a linear combination of (*i.e.*, regression on) the current input and some previous values of the output.

Another widely used model is the *autoregressive moving average (ARMA) model*, which combines the MA and AR models:

$$y(k) = b_1 y(k-1) + \cdots + b_p y(k-p) + a_0 u(k) + \cdots + a_r u(k-r).$$

Finally, the problem: Express each of these models as a linear dynamical system with input u and output y. For the MA model, use state

$$x(k) = \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-r) \end{bmatrix},$$

and for the AR model, use state

$$x(k) = \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-p) \end{bmatrix}.$$

You decide on an appropriate state vector for the ARMA model. (There are many possible choices for the state here, even with different dimensions. We recommend you choose a state for the ARMA model that makes it easy for you to derive the state equations.)

Remark: multi-input, multi-output time-series models (*i.e.*, $u(k) \in \mathbf{R}^m$, $y(k) \in \mathbf{R}^p$) are readily handled by allowing the coefficients a_i, b_i to be matrices.

Solution:
In this problem we should find matrices A, B, C and D such that

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k). \end{aligned}$$

- *Moving average model.* We need to express $x(k+1)$ linearly in terms of $x(k)$ and $u(k)$. We have

$$x(k) = \begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(k-r) \end{bmatrix}$$

and therefore

$$x(k+1) = \begin{bmatrix} u(k) \\ u(k-1) \\ \vdots \\ u(k+1-r) \end{bmatrix}.$$

Note that

$$x(k+1) = \begin{bmatrix} 0 \\ u(k-1) \\ \vdots \\ u(k+1-r) \end{bmatrix} + \begin{bmatrix} u(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

but

$$\begin{bmatrix} 0 \\ u(k-1) \\ \vdots \\ u(k+1-r) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(k-r) \end{bmatrix}}_{x(k)}$$

so

$$x(k+1) = \underbrace{\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{A} x(k) + \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{B} u(k).$$

$y(k)$ should be expressed in terms of $x(k)$ and $u(k)$. This is easy from the relation $y(k) = a_0u(k) + a_1u(k-1) + \cdots + a_ru(k-r)$ and we get

$$y(k) = \underbrace{\begin{bmatrix} a_1 & a_2 & \cdots & a_r \end{bmatrix}}_{C} x(k) + \underbrace{a_0}_{D} u(k).$$

(Note: the matrix A with ones on its subdiagonal is called a *shift matrix* because it shifts down the elements of the input vector.)

- *Autoregressive model.* In this case

$$x(k) = \begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-p) \end{bmatrix}$$

so

$$x(k+1) = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k+1-p) \end{bmatrix} = \begin{bmatrix} 0 \\ y(k-1) \\ \vdots \\ y(k+1-p) \end{bmatrix} + \begin{bmatrix} y(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Now

$$\begin{bmatrix} 0 \\ y(k-1) \\ \vdots \\ y(k+1-p) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-p) \end{bmatrix}}_{x(k)}$$

and

$$\begin{bmatrix} y(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & \cdots & b_p \\ 0 & 0 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 0 \end{bmatrix} \underbrace{\begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-p) \end{bmatrix}}_{x(k)} + \begin{bmatrix} u(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Thus

$$x(k+1) = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} b_1 & b_2 & \cdots & b_p \\ 0 & 0 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(k)$$

or

$$x(k+1) = \underbrace{\begin{bmatrix} b_1 & b_2 & b_3 & \cdots & b_p \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{A} x(k) + \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{B} u(k)$$

and

$$y(k) = \underbrace{\begin{bmatrix} b_1 & b_2 & \cdots & b_p \end{bmatrix}}_{C} x(k) + \underbrace{1}_{D} u(k).$$

- *Autoregressive moving average model.* One simple choice for $x(k)$ is

$$x(k) = \begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(k-r) \\ \hline y(k-1) \\ y(k-2) \\ \vdots \\ y(k-p) \end{bmatrix}$$

and therefore

$$x(k+1)=\left[\begin{array}{c}0\\u(k-1)\\\vdots\\u(k+1-r)\\\hline 0\\0\\\vdots\\0\end{array}\right]+\left[\begin{array}{c}u(k)\\0\\\vdots\\0\\\hline 0\\0\\\vdots\\0\end{array}\right]+\left[\begin{array}{c}0\\0\\\vdots\\0\\\hline y(k)\\y(k-1)\\\vdots\\y(k+1-p)\end{array}\right].$$

For similar reasons to the previous parts

$$\left[\begin{array}{c}0\\u(k-1)\\u(k-2)\\\vdots\\u(k+1-r)\\\hline 0\\\vdots\\0\end{array}\right]=\left[\begin{array}{ccccc|ccccc}0&0&0&\cdots&0&0&0&0&\cdots&0\\1&0&0&\cdots&0&\vdots&&&&\vdots\\0&1&0&\cdots&0&\vdots&&&&\vdots\\\vdots&&\ddots&\ddots&\vdots&\vdots&&\ddots&\ddots&0\\0&0&\cdots&1&0&0&0&\cdots&0&0\\\hline 0&0&0&\cdots&0&0&0&0&\cdots&0\\\vdots&&&&\vdots&\vdots&&&&\vdots\\0&0&0&\cdots&0&0&0&0&\cdots&0\end{array}\right]x(k)$$

and

$$\left[\begin{array}{c}0\\\vdots\\0\\\hline y(k)\\y(k-1)\\y(k-2)\\\vdots\\y(k+1-p)\end{array}\right]=\left[\begin{array}{ccccc|ccccc}0&0&&\cdots&0&0&0&0&\cdots&0\\0&&&\cdots&0&\vdots&&&&\vdots\\0&&&\cdots&0&0&0&&\cdots&0\\\hline a_1&a_2&a_3&\cdots&a_r&b_1&b_2&b_3&\cdots&b_p\\0&&&\cdots&0&1&0&0&\cdots&0\\0&&&\cdots&0&0&1&0&\cdots&0\\\vdots&&\ddots&\ddots&\vdots&&\ddots&\ddots&\vdots&\\0&&\cdots&0&0&0&0&\cdots&1&0\end{array}\right]x(k)+\left[\begin{array}{c}0\\\vdots\\0\\\hline a_0\\0\\0\\\vdots\\0\end{array}\right]u(k).$$

Thus

$$x(k+1) = \left[\begin{array}{ccccc|ccccc}
0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\
1 & 0 & 0 & \cdots & 0 & \vdots & & & & \vdots \\
0 & 1 & 0 & \cdots & 0 & \vdots & & & & \vdots \\
\vdots & & \ddots & \ddots & \vdots & \vdots & & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & 0 & & \cdots & 0 \\
\hline
a_1 & a_2 & a_3 & \cdots & a_r & b_1 & b_2 & b_3 & \cdots & b_p \\
0 & & & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\
0 & & & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\
\vdots & & \ddots & \ddots & \vdots & & \ddots & \ddots & \vdots & \\
0 & & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0
\end{array}\right] x(k) + \left[\begin{array}{c}
1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \hline a_0 \\ 0 \\ 0 \\ \vdots \\ 0
\end{array}\right] u(k)$$

and

$$y(k) = \underbrace{\left[\begin{array}{cccc|cccc} a_1 & a_2 & \cdots & a_r & b_1 & b_2 & \cdots & b_p \end{array}\right]}_{C} x(k) + \underbrace{a_0}_{D} u(k).$$

(Note: it is possible to give state-space models with a fewer number of states but this is not our concern here. A state-space model for the system with the fewest number of states is called a *minimal realization* for the system. This topic will be covered later in the course.)

2.4 *Representing linear functions as matrix multiplication.* Suppose that $f : \mathbf{R}^n \longrightarrow \mathbf{R}^m$ is linear. Show that there is a matrix $A \in \mathbf{R}^{m\times n}$ such that for all $x \in \mathbf{R}^n$, $f(x) = Ax$. (Explicitly describe how you get the coefficients A_{ij} from f, and then verify that $f(x) = Ax$ for any $x \in \mathbf{R}^n$.)

Is the matrix A that represents f unique? In other words, if $\tilde{A} \in \mathbf{R}^{m\times n}$ is another matrix such that $f(x) = \tilde{A}x$ for all $x \in \mathbf{R}^n$, then do we have $\tilde{A} = A$? Either show that this is so, or give an explicit counterexample.

Solution:
Any $x = \left[\begin{array}{cccc} x_1 & x_2 & \cdots & x_n \end{array}\right]^T \in \mathbf{R}^n$ can be written as

$$x = x_1 e_1 + x_2 e_2 + \cdots + x_n e_n$$

where e_i is the ith unit vector in $\mathbf{R}^n$. From linearity of $f(\cdot)$ we have

$$f(x) = x_1 f(e_1) + x_2 f(e_2) + \cdots + x_n f(e_n),$$

or in (block) matrix form

$$f(x) = \left[\begin{array}{cccc} f(e_1) & f(e_2) & \cdots & f(e_n) \end{array}\right] x.$$

Therefore, we simply take

$$A := \left[\begin{array}{cccc} f(e_1) & f(e_2) & \cdots & f(e_n) \end{array}\right].$$

In other words, we only need the transformations of the unit vectors e_i to form the matrix A. Note that $f(e_i) \in \mathbf{R}^m$ for $i = 1, 2, \ldots, n$ so A becomes an $m \times n$ matrix.

Suppose the matrix A is not unique and there is another $\tilde{A} \in \mathbf{R}^{m\times n}$ such that $f(x) = \tilde{A}x$. Then $Ax = \tilde{A}x$ or $(A - \tilde{A})x = 0$ for all $x \in \mathbf{R}^n$. When $x = e_i$, $(A - \tilde{A})e_i = 0$ implies that the ith column of $(A - \tilde{A})$ is zero. Repeating this argument for $i = 1, 2, \ldots, n$ proves that *all* columns of $(A - \tilde{A})$ are zero and hence $A = \tilde{A}$. Therefore the choice of A is *unique*.

2.6 *Matrix representation of polynomial differentiation.* We can represent a polynomial of degree less than n,

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0,$$

as the vector $[a_0\ a_1\ \cdots\ a_{n-1}]^T \in \mathbf{R}^n$. Consider the linear transformation $\mathcal{D}$ that differentiates polynomials, *i.e.*, $\mathcal{D}p = dp/dx$. Find the matrix D that represents $\mathcal{D}$ (*i.e.*, if the coefficients of p are given by a, then the coefficients of dp/dx are given by Da).

Solution:

According to problem 2.4, it suffices to compute the transformation of the unit vectors $e_i \in \mathbf{R}^n$ for $i = 1, 2, \ldots, n$ under differentiation. In other words

$$D = \begin{bmatrix} De_1 & De_2 & \cdots & De_n \end{bmatrix}.$$

e_1 corresponds to the polynomial $p_1(x) = 1$, and since $\mathcal{D}p_1 = 0$ we have

$$De_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

e_2 corresponds to $p_2(x) = x$ with $\mathcal{D}p_2 = 1$ and therefore

$$De_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = e_1.$$

Similarly for $p_3(x) = x^2$ we get $\mathcal{D}p_3 = 2x$ so

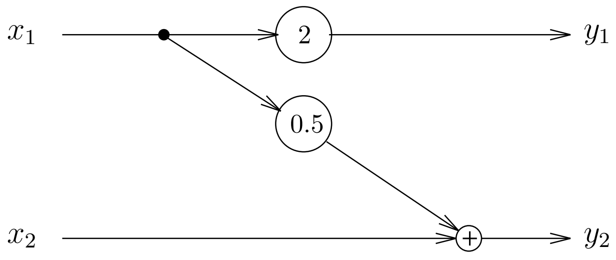
$$De_3 = \begin{bmatrix} 0 \\ 2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 2e_2.$$

It is easy to see that, in general, for $i > 1$ we have $De_i = (i-1)e_{i-1}$. Therefore

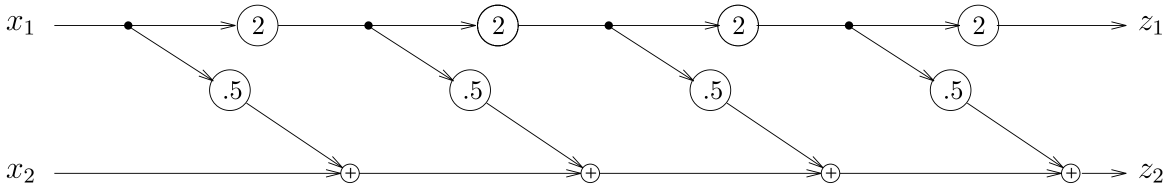
$$D = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 3 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & n-1 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

2.9 *Matrices and signal flow graphs.*

(a) Find $A \in \mathbf{R}^{2\times 2}$ such that $y = Ax$ in the system below:



(b) Find $B \in \mathbf{R}^{2\times 2}$ such that $z = Bx$ in the system below:

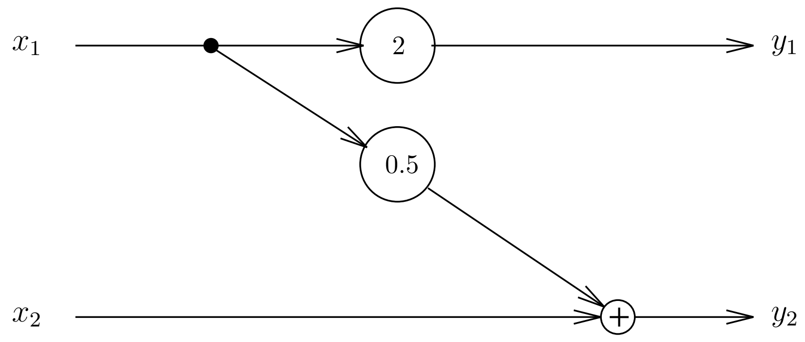


Do this two ways: first, by expressing the matrix B in terms of A from the previous part (explaining why they are related as you claim); and second, by directly evaluating all possible paths from each x_j to each z_i.

Solution:

(a) By evaluating path gains we have

- *Gain from x_1 to y_1.* There is only one path with gain 2.
- *Gain from x_1 to y_2.* There is only one path with gain 0.5.
- *Gain from x_2 to y_1.* There are no paths and therefore the gain is 0.
- *Gain from x_2 to y_2.* There is only one path with gain 1.



and therefore

$$A = \begin{bmatrix} 2 & 0 \\ 0.5 & 1 \end{bmatrix}.$$

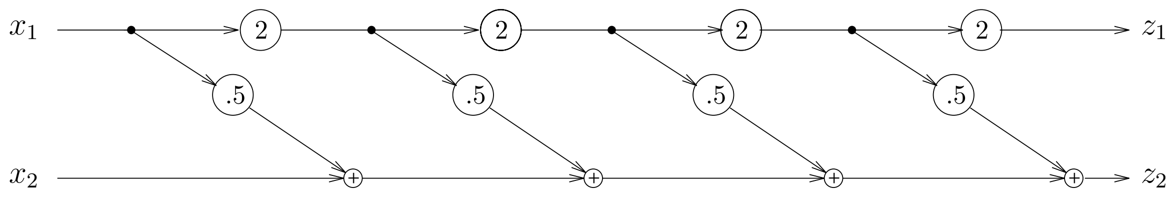
(b) Clearly $B = A^4$. Carrying out the multiplication gives

$$B = \begin{bmatrix} 16 & 0 \\ 7.5 & 1 \end{bmatrix}.$$

Now by directly evaluating all possible path gains we get

- *Gain from x_1 to z_1.* There is only one path with gain $2 \times 2 \times 2 \times 2 = 16$
- *Gain from x_1 to z_2.* There are 4 possible paths. These paths have gains 0.5, 2×0.5, $2 \times 2 \times 0.5$ and $2 \times 2 \times 2 \times 0.5$ that sum up to 7.5.
- *Gain from x_2 to z_1.* There are no paths and therefore the gain is 0.
- *Gain from x_2 to z_2.* There is only one path with gain 1.

and therefore we get the same B as expected.



2.12 *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 $A = A^T$, 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 $k \in \mathbf{Z}$, $k \geq 1$. 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 $m_1, m_2, \ldots, m_{k-1}$.

Solution to additional exercise

1. *Affine functions.* A function $f : \mathbf{R}^n \to \mathbf{R}^m$ is called *affine* if for any $x,\ y \in \mathbf{R}^n$ and any $\alpha,\ \beta \in \mathbf{R}$ 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 $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$. 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 $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$. (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 $x \in \mathbf{R}^n$ and any $\alpha \in \mathbf{R}$.

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

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 $A \in \mathbf{R}^{m\times n}$ 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 $f(x) = Ax + b$.