

Homework 1 Solutions

Due Thursday 10/1.

1. *Interpretations of matrix multiplication.*

Let B be a 4×4 matrix, to which we apply the following operations:

- (i) double column 1,
 - (ii) halve row 3,
 - (iii) add row 3 to row 1,
 - (iv) interchange columns 1 and 4,
 - (v) subtract row 2 from each of the other rows,
 - (vi) replace column 4 by column 3,
 - (vii) delete column 1 (so that the column dimension is reduced by 1)
- (a) Write the result as a product of 8 matrices.
 (b) Write it again as a product ABC (same B) of three matrices.

Solution.

(a) Each of the operations can be performed as follows.

(a) Double column 1: let B be

$$B = [b_1 \quad b_2 \quad b_3 \quad b_4]$$

where b_i is a column vector in R^4 . Then postmultiply by a diagonal matrix as follows.

$$H = [b_1 \quad b_2 \quad b_3 \quad b_4] \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_W = [2b_1 \quad b_2 \quad b_3 \quad b_4]$$

(b) Halve row 3: let the rows of H be

$$H = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \\ h_4^T \end{bmatrix}$$

then

$$J = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_P \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \\ h_4^T \end{bmatrix} = \begin{bmatrix} h_1^T \\ h_2^T \\ \frac{1}{2}h_3^T \\ h_4^T \end{bmatrix}$$

(c) Add row 3 to row 1: premultiply J as follows.

$$K = \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_Q \begin{bmatrix} j_1^T \\ j_2^T \\ j_3^T \\ j_4^T \end{bmatrix} = \begin{bmatrix} j_1^T + j_3^T \\ j_2^T \\ j_3^T \\ j_4^T \end{bmatrix}$$

(d) Interchange column 1 and 4

$$L = [k_1 \quad k_2 \quad k_3 \quad k_4] \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_R = [k_4 \quad k_2 \quad k_3 \quad k_1]$$

(e) Subtract row 2 from each of the other rows

$$M = \underbrace{\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}}_S \begin{bmatrix} l_1^T \\ l_2^T \\ l_3^T \\ l_4^T \end{bmatrix} = \begin{bmatrix} l_1^T - l_2^T \\ l_2^T \\ l_3^T - l_2^T \\ l_4^T - l_2^T \end{bmatrix}$$

(f) Replace column 4 by column 3

$$N = [m_1 \quad m_2 \quad m_3 \quad m_4] \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_T = [m_1 \quad m_2 \quad m_3 \quad m_3]$$

(g) Delete column 1 (so that the column dimension is reduced by 1)

$$A = [n_1 \quad n_2 \quad n_3 \quad n_4] \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_U = [n_2 \quad n_3 \quad n_4]$$

Then we have the result as the following product of 8 matrices

$$A = SQPBWRTU$$

(b) Write it again as a product ABC (same B) of three matrices.

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

2. The final position/velocity of a mass.

Suppose we apply a sequence of 10 force inputs to a unit mass, with the i 'th force given by x_i , as in the example on slide 2-3. Let $y \in \mathbb{R}^2$ be a vector with

$$y_1 = \text{position of mass at time 10}$$

$$y_2 = \text{velocity of mass at time 10}$$

Find the 2×10 matrix A so that $y = Ax$.

Find any sequence of force inputs x which transfer the mass to position 2 with final velocity 0.

Now suppose we want to maximize the final position, (we don't care about the final velocity), using a sequence of forces which satisfy

$$\sum_{i=1}^{10} x_i^2 \leq 1$$

Find an optimum force input sequence? Is it unique?

Solution.

Suppose $q(t), v(t)$ are the position and velocity of the mass at time t . Applying Newton's law gives

$$\begin{aligned} q(t+1) &= q(t) + v(t) + \frac{1}{2}x_{t+1} \\ v(t+1) &= v(t) + x_{t+1} \end{aligned}$$

Iterating this gives

$$\begin{aligned} q(1) &= \frac{1}{2}x_1 & q(2) &= \frac{3}{2}x_1 + \frac{1}{2}x_2 \\ v(1) &= x_1 & v(2) &= x_1 + x_2 \end{aligned}$$

The pattern continues, until we arrive at

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}}_y = \underbrace{\begin{bmatrix} \frac{19}{2} & \frac{17}{2} & \frac{15}{2} & \frac{13}{2} & \frac{11}{2} & \frac{9}{2} & \frac{7}{2} & \frac{5}{2} & \frac{3}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix}}_x$$

We are looking for x such that

$$y = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

If we set $x_i = 0$ for $i = 1, \dots, 8$ then we have

$$y = \begin{bmatrix} \frac{3}{2} & \frac{1}{2} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \end{bmatrix}$$

hence choosing

$$\begin{bmatrix} x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & \frac{1}{2} \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

gives a suitable force input as

$$x = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 2 \ -2]$$

There are many other possible solutions.

Since we don't care about the final velocity, we can ignore the second row of this matrix equation. Then

$$y_1 = \underbrace{\begin{bmatrix} \frac{19}{2} & \frac{17}{2} & \frac{15}{2} & \frac{13}{2} & \frac{11}{2} & \frac{9}{2} & \frac{7}{2} & \frac{5}{2} & \frac{3}{2} & \frac{1}{2} \end{bmatrix}}_{a^T} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix}$$

Since a is a vector in \mathbb{R}^{10} , we have $a^T x$ is the component of x in the direction of a , scaled by the norm of a . If a and x are in the same direction, $a^T x$ will take its the maximum value. So the optimum force input sequence is

$$x = \frac{a}{\|a\|} = \frac{1}{18.2346} \left[\frac{19}{2} \quad \frac{17}{2} \quad \frac{15}{2} \quad \frac{13}{2} \quad \frac{11}{2} \quad \frac{9}{2} \quad \frac{7}{2} \quad \frac{5}{2} \quad \frac{3}{2} \quad \frac{1}{2} \right]^T$$

and it is unique.

3. *Some standard time-series models.*

A time series is just a discrete-time signal, *i.e.*, a function from \mathbf{Z}_+ into \mathbb{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*)

$$y(k) = a_0 u(k) + a_1 u(k-1) + \cdots + a_r u(k-r)$$

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.*, $u(k) \in \mathbb{R}^m$, $y(k) \in \mathbb{R}^p$) 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 $y(k) = a_0 u(k) + a_1 u(k-1) + \cdots + a_r u(k-r)$ 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

$$y(k) = \underbrace{[b_1 \ b_2 \ \cdots \ b_p]}_C x(k) + \underbrace{1}_D u(k).$$

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

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

and therefore

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

For similar reasons to the previous parts

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

and

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

Thus

$$x(k+1) = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & | & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 & | & \vdots & & & & \vdots \\ 0 & 1 & 0 & \cdots & 0 & | & \vdots & & & & \vdots \\ \vdots & & \ddots & \ddots & \vdots & | & \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & | & 0 & 0 & \cdots & 0 & \\ \hline a_1 & a_2 & a_3 & \cdots & a_r & | & b_1 & b_2 & b_3 & \cdots & b_p \\ 0 & & \cdots & 0 & | & 1 & 0 & 0 & \cdots & 0 \\ 0 & & \cdots & 0 & | & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots & & \ddots & \ddots & \vdots & & \\ 0 & & \cdots & 0 & 0 & | & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \hline a_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(k)$$

and

$$y(k) = \underbrace{[a_1 \ a_2 \ \cdots \ a_r]}_C x(k) + \underbrace{a_0}_D u(k).$$

(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.)

4. Representing linear functions as matrix multiplication.

Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is linear. Show that there is a matrix $A \in \mathbb{R}^{m \times n}$ such that for all $x \in \mathbb{R}^n$, $f(x) = Ax$. (Explicitly describe how you get the coefficients A_{ij} from f , and then verify that $f(x) = Ax$ for any $x \in \mathbb{R}^n$.) Is the matrix A that represents f unique? In other words, if $\tilde{A} \in \mathbb{R}^{m \times n}$ is another matrix such that $f(x) = \tilde{A}x$ for all $x \in \mathbb{R}^n$, then do we have $\tilde{A} = A$? Either show that this is so, or give an explicit counterexample.

Solution.

Any $x = [x_1 \ x_2 \ \cdots \ x_n]^T \in \mathbb{R}^n$ can be written as

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

where e_i is the i th unit vector in \mathbb{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

$$f(x) = [f(e_1) \ f(e_2) \ \cdots \ f(e_n)] x.$$

Therefore, we simply take

$$A := [f(e_1) \ f(e_2) \ \cdots \ f(e_n)].$$

In other words, we only need the transformations of the unit vectors e_i to form the matrix A . Note that $f(e_i) \in \mathbb{R}^m$ for $i = 1, 2, \dots, n$ so A becomes an $m \times n$ matrix. Suppose the matrix A is not unique and there is another $\tilde{A} \in \mathbb{R}^{m \times n}$ such that $f(x) = \tilde{A}x$. Then $Ax = \tilde{A}x$ or $(A - \tilde{A})x = 0$ for all $x \in \mathbb{R}^n$. When $x = e_i$, $(A - \tilde{A})e_i = 0$ implies that the i th column of $(A - \tilde{A})$ is zero. Repeating this argument for $i = 1, 2, \dots, n$ proves that *all* columns of $(A - \tilde{A})$ are zero and hence $A = \tilde{A}$. Therefore the choice of A is *unique*.

5. 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 $t = 0, 1, 2, \dots$. 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 i th 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

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

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:

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

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

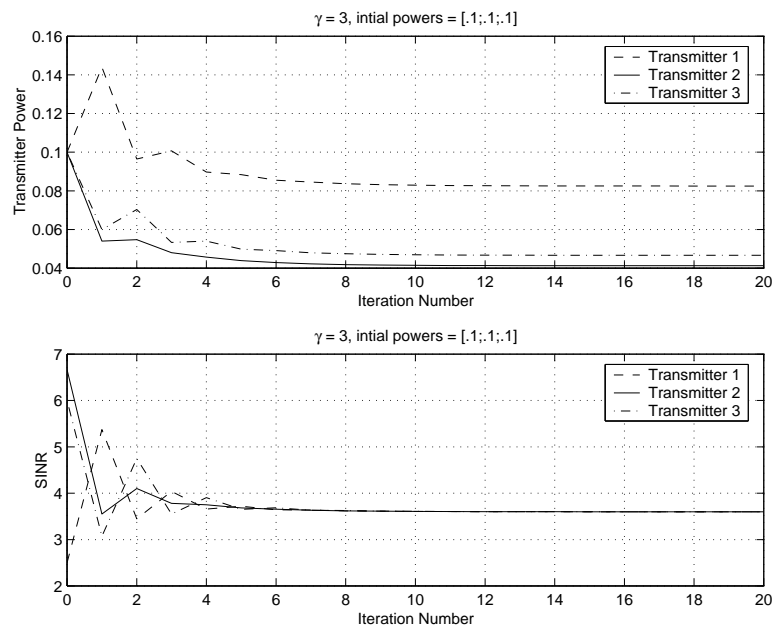
```
clear all; close all;
G = [1 .2 .1; .1 2 .1; .3 .1 3];% Gain matrix
gamma = 3; % minimum SINR
alpha = 1.2; % safety margin
sigma = 0.01; % Noise power (same for all receivers)
A = zeros(3,3); for i = 1:3
for j = 1:3
if (i~=j)
A(i,j) = alpha*gamma*G(i,j)/G(i,i);
end
end
end
b = zeros(3,1); for i =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

```

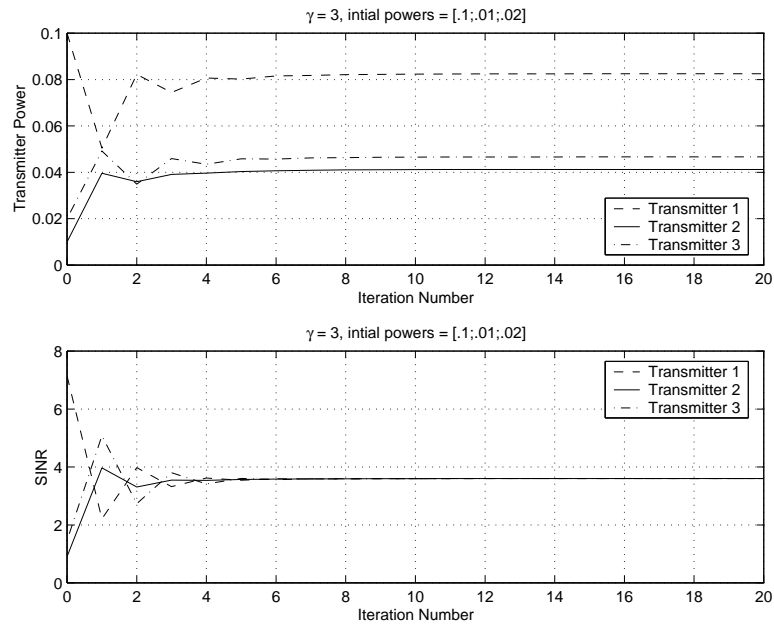
p_i = A*p_i+b;
p = [p p_i]; % Find the new powers and save
SINR_current = [G(1,1)*p_i(1)/(sigma+G(1,2)*p_i(2)+G(1,3)*p_i(3)); \
G(2,2)*p_i(2)/(sigma+G(2,1)*p_i(1)+G(2,3)*p_i(3)); \
G(3,3)*p_i(3)/(sigma+G(3,1)*p_i(1)+G(3,2)*p_i(2))];
S = [S SINR_current];
end
figure(1); temp = 0:num_iterations; subplot(2,1,1);
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;

```

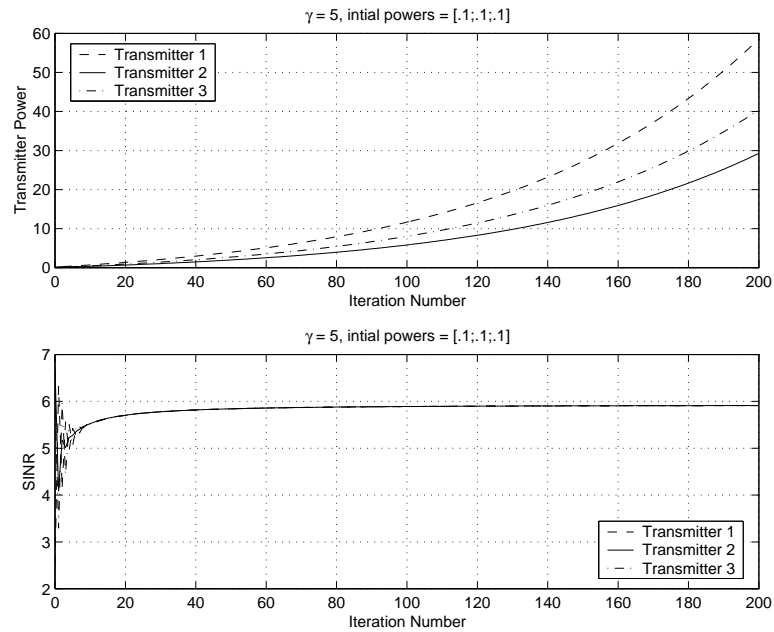
The figure below shows the SINR and transmitter power as a function of iteration number.

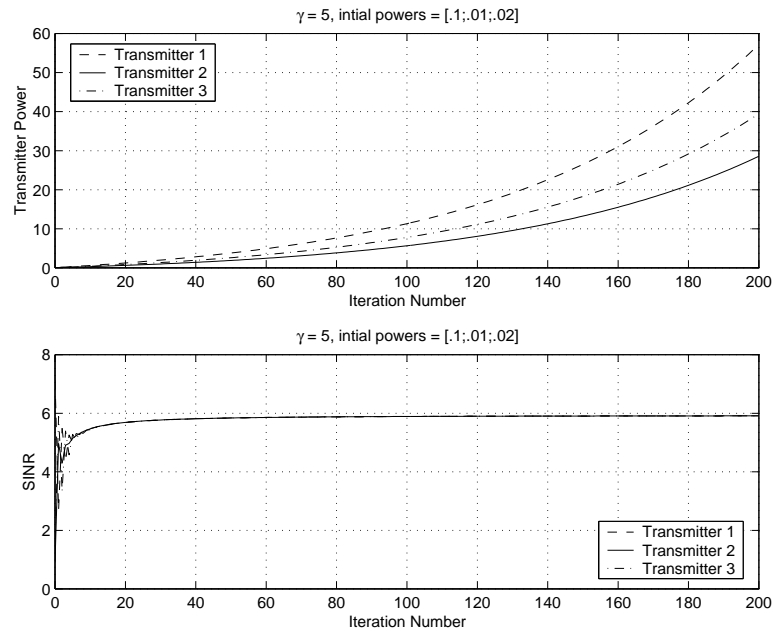


Similar matlab code can be used to try other initial transmitter powers. For example, the simulation shown below used initial transmitter powers of .1, .01, and .02 for the first, second, and third transmitter respectively. In both cases, the final transmitter powers approach .083, .041, and .047. The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



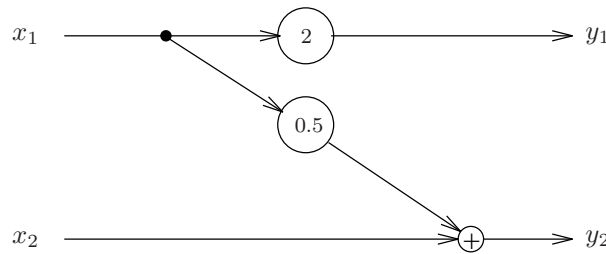
Testing the system for $\gamma = 5$ and the same initial conditions (see graphs below) shows that the algorithm does not always succeed. For both initial conditions tried, the transmitter powers grow exponentially. Also, the SINR approaches $5.92 < \alpha\gamma = 6$.



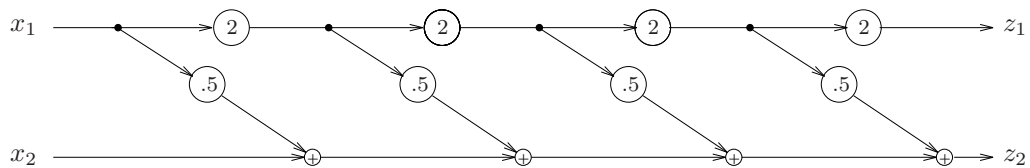


6. Matrices and signal flow graphs.

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



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

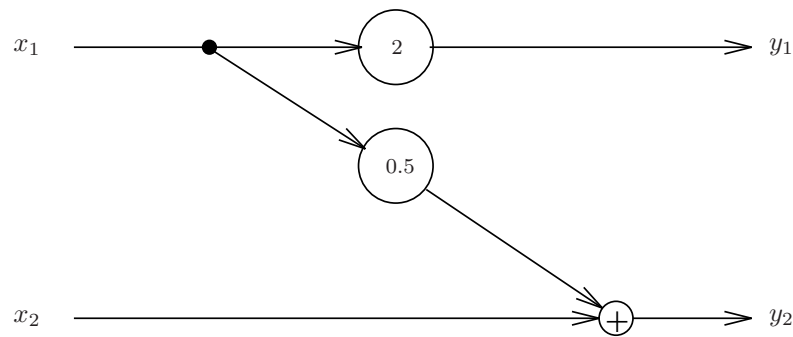


Do this two ways: first, by expressing the matrix B in terms of A from the previous part (explaining why they are related as you claim); and second, by directly evaluating all possible paths from each x_j to each z_i .

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

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

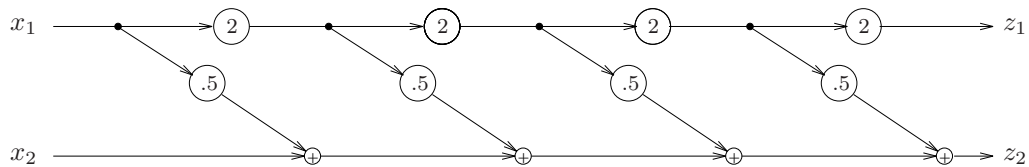
(b) Clearly $B = A^4$. Carrying out the multiplication gives

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

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.



7. Counting sequences in a language or code.

We consider a language or code with an alphabet of n symbols $1, 2, \dots, n$. A sentence is a finite sequence of symbols, k_1, \dots, k_m where $k_i \in \{1, \dots, n\}$. 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

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

- Let $B = A^k$. Give an interpretation of B_{ij} in terms of the language.
- 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) This problem is similar to, but not quite the same as, the problem on counting paths of a given length. 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 .
- (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 . From 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 ...)