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 $\gamma$ (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 $i$th transmitter) increases $S_i$ but decreases all other $S_j$. A very simple power update algorithm is given by

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

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 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 $\sigma, \gamma, \alpha$ and the components of $G$. 
b) simulation Simulate the power control algorithm, starting from various initial (positive) power levels. Use the problem data

\[
G = \begin{bmatrix}
1 & 0.2 & 0.1 \\
0.1 & 2 & 0.1 \\
0.3 & 0.1 & 3
\end{bmatrix}, \quad \gamma = 3, \quad \alpha = 1.2, \quad \sigma = 0.1.
\]

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.

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

In matrix form the equations look like this:

\[
\begin{bmatrix}
p_1(t + 1) \\
p_2(t + 1) \\
p_3(t + 1) \\
\vdots \\
p_n(t + 1)
\end{bmatrix} =
\begin{bmatrix}
0 & \alpha \gamma G_{12} & \alpha \gamma G_{13} & \cdots & \alpha \gamma G_{1n} \\
\alpha \gamma G_{21} & 0 & \alpha \gamma G_{23} & \cdots & \alpha \gamma G_{2n} \\
\alpha \gamma G_{31} & \alpha \gamma G_{32} & 0 & \cdots & \alpha \gamma G_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha \gamma G_{n1} & \alpha \gamma G_{n2} & \alpha \gamma G_{n3} & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
p_1(t) \\
p_2(t) \\
p_3(t) \\
\vdots \\
p_n(t)
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
\alpha \gamma \sigma^2 \\
\frac{\alpha \gamma \sigma^2}{G_{11}} \\
\frac{\alpha \gamma \sigma^2}{G_{21}} \\
\vdots \\
\frac{\alpha \gamma \sigma^2}{G_{nn}}
\end{bmatrix}
\]

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

```matlab
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
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
```
\[ b \equiv \text{zeros}(3,1); \text{for } i \equiv 1:3 \]
\[ b(i) = \alpha \gamma \sigma / G(i,i); \]
end
\[ \text{num\_iterations} = 20; \]
\[ p_{i} = [.1; .1; .1]; \% \text{Initialized to } p(0) \]
\[ S = \begin{bmatrix} G(1,1) p_{i}(1)/(\sigma + G(1,2) p_{i}(2) + G(1,3) p_{i}(3)); \\
G(2,2) p_{i}(2)/(\sigma + G(2,1) p_{i}(1) + G(2,3) p_{i}(3)); \\
G(3,3) p_{i}(3)/(\sigma + G(3,1) p_{i}(1) + G(3,2) p_{i}(2)) \end{bmatrix}; \]
p = p_{i}; \% \text{matrix to store the powers versus time}
\[ \text{for } i = 1: \text{num\_iterations} \]
p_{i} = A p_{i} + b;
p = [p p_{i}]; \% \text{Find the new powers and save}
\[ \text{SINR\_current} = \begin{bmatrix} G(1,1) p_{i}(1)/(\sigma + G(1,2) p_{i}(2) + G(1,3) p_{i}(3)); \\
G(2,2) p_{i}(2)/(\sigma + G(2,1) p_{i}(1) + G(2,3) p_{i}(3)); \\
G(3,3) p_{i}(3)/(\sigma + G(3,1) p_{i}(1) + G(3,2) p_{i}(2)) \end{bmatrix}; \]
\[ S = [S \text{ SINR\_current}]; \]
end
\[ \text{figure(1); temp} = 0: \text{num\_iterations}; \text{subplot}(2,1,1); \]
\[ \text{plot}(\text{temp},p(1,:),'--', \text{temp},p(2,:),'-', \text{temp},p(3,:),'.-'); \]
\[ \text{xlabel('Iteration Number'); ylabel('Transmitter Power'); title('\gamma = 3, initial powers = [.1; .1; .1]');} \]
\[ \text{legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;} \]
\[ \text{subplot}(2,1,2); \text{plot}(\text{temp},S(1,:),'-', \text{temp},S(2,:),'-', \text{temp},S(3,:),'-'); \text{xlabel('Iteration Number');} \]
\[ \text{ylabel('SINR'); title('\gamma = 3, initial powers = [.1; .1; .1]');} \]
\[ \text{legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;} \]

The figure below shows the SINR and transmitter power as a function of iteration num-
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 trans-
mitter powers grow exponentially. Also, the SINR approaches $5.92 < \alpha \gamma = 6$.

2. 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 \( x_1, \ldots, x_4 \) 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:

\[
v(t) = v(0) + \int_0^t f(\tau) d\tau \\
= v(0) + \sum_{j=1}^{\lfloor t \rfloor} \int_{j-1}^j f(\tau) d\tau + \int_{\lfloor t \rfloor}^t f(\tau) d\tau \\
= v(0) + \sum_{j=1}^{\lfloor t \rfloor} \int_{j-1}^j x_j d\tau + \int_{\lfloor t \rfloor}^t x_{\lfloor t \rfloor + 1} d\tau \\
= v(0) + \sum_{j=1}^{\lfloor t \rfloor} (\tau x_j)_{\tau=j-1}^{\tau=j} + (\tau x_{\lfloor t \rfloor + 1})_{\tau=\lfloor t \rfloor}^{\tau=t} \\
= v(0) + \sum_{j=1}^{\lfloor t \rfloor} x_j + (t - \lfloor t \rfloor)x_{\lfloor t \rfloor + 1}.
\]

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:

\[
p(t) = p(0) + \int_0^t v(\tau) d\tau \\
= p(0) + \int_0^t (v(0) + (v(\tau) - v(0))) d\tau \\
= p(0) + v(0)t + \int_0^t (v(\tau) - v(0)) d\tau \\
= p(0) + v(0)t + \sum_{j=1}^{\lfloor \tau \rfloor} \int_{j-1}^j (v(\tau) - v(0)) d\tau + \int_{\lfloor \tau \rfloor}^t (v(\tau) - v(0)) d\tau \\
= p(0) + v(0)t + \sum_{j=1}^{\lfloor \tau \rfloor} \int_{j-1}^j \left( \sum_{k=1}^{\lfloor \tau \rfloor} x_k + (\tau - \lfloor \tau \rfloor)x_{\lfloor \tau \rfloor + 1} \right) d\tau
\]
Thus, we obtain the following system of linear equations:

\[
\begin{align*}
&+ \int_{[t]} \left( \sum_{k=1}^{[\tau]} x_k + (\tau - [\tau])x_{[\tau]} + 1 \right) d\tau \\
&= p(0) + v(0) t + \sum_{j=1}^{[\tau]} \int_{j-1}^{j} \left( \sum_{k=1}^{j-1} x_k + (\tau - (j-1))x_j \right) d\tau \\
&+ \int_{[t]} \left( \sum_{k=1}^{[\tau]} x_k + (\tau - [\tau])x_{[\tau]} + 1 \right) d\tau \\
&= p(0) + v(0) t + \sum_{k=1}^{[\tau]} \left( \sum_{k=1}^{j-1} x_k + 1/2 (\tau - (j-1))^2 x_j + \left( \sum_{k=1}^{[\tau]} x_k + 1/2 (t - [t])^2 x_{[t]+1} \right) \right)
\end{align*}
\]

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

\[
y_1 = p(n) = \sum_{j=1}^{n} (n - k + \frac{1}{2}) x_j.
\]

Thus, we obtain the following system of linear equations:

\[
y_1 = \sum_{j=1}^{n} (n - j + \frac{1}{2}) x_j,
\]

\[
y_2 = \sum_{j=1}^{n} x_j.
\]
Since $A_{ij}$ gives the coefficient of $x_j$ in our expression for $y_i$, we have that
\[ A_{1j} = n - j + \frac{1}{2} \quad \text{and} \quad A_{2j} = 1, \quad j = 1, \ldots, n. \]

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:
\[
\begin{bmatrix}
7 & 5 & 3 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix}
=
\begin{bmatrix}
1 \\
0
\end{bmatrix}.
\]

This system is underdetermined, and has infinitely many solutions. Suppose we choose $x_2 = x_3 = 0$. Then, we are left with the system
\[
\begin{bmatrix}
7 & 1 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_4
\end{bmatrix}
=
\begin{bmatrix}
1 \\
0
\end{bmatrix}.
\]

The second equation implies that $x_4 = -x_1$. Then, the first equation becomes
\[
\frac{7}{2}x_1 + \frac{1}{2}x_4 = \frac{7}{2}x_1 - \frac{1}{2}x_1 = 3x_1 = 1.
\]

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

3. 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 $y = q$. 9
**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:

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

4. **Express the following statements in matrix language.** You can assume that all matrices mentioned have appropriate dimensions. Here is an example: “Every column of \( C \) is a linear combination of the columns of \( B \)” can be expressed as “\( C = BF \) for some matrix \( F \)”.

There can be several answers; one is good enough for us.

a) Suppose \( Z \) has \( n \) columns. For each \( i \), row \( i \) of \( Z \) is a linear combination of rows \( i, \ldots, n \) of \( Y \).

b) \( W \) is obtained from \( V \) by permuting adjacent odd and even columns (i.e., 1 and 2, 3 and 4, \ldots).

c) Each column of \( P \) makes an acute angle with each column of \( Q \).

d) Each column of \( P \) makes an acute angle with the corresponding column of \( Q \).

e) The first \( k \) columns of \( A \) are orthogonal to the remaining columns of \( A \).
Solution.

a) $Z = UY$, where $U$ is upper triangular, i.e., $U_{ij} = 0$ for $i > j$.

b) $W = VS$, where $S$ is the odd-even switch matrix, defined as 

$$S = \begin{bmatrix} e_2 & e_1 & e_4 & e_3 & \cdots & e_m & e_{m-1} \end{bmatrix}.$$ 

c) All entries of the matrix $P^TQ$ are positive.

d) The diagonal entries of the matrix $P^TQ$ are positive.

e) $A^T A$ has the block diagonal form 

$$A^T A = \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix},$$

where $B_{11} \in \mathbb{R}^{k \times k}$.

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 must be followed by 1 or 3; symbol 2 must be followed by 2 or 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.
c) Consider the Markov language of part (b), among all allowed sequences of length 10, find the most common value for the seventh symbol. In principle you could solve this problem by writing down all allowed sequences of length 10, and counting how many of these have symbol \( i \) as the seventh symbol, for \( i = 1, \ldots, 5 \). (We’re interested in the symbol for which this count is largest.) But we’d like you to use a smarter approach. Explain clearly how you solve the problem, as well as giving the specific answer. Hint: you may find the interpretation of \( A^k \) helpful.

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:

\[
S_{p+2}(i, j) = \bigcup_{k \in \phi(i)} (S_{p+1}(k, j) + i).
\]

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

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

12
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 $\text{sum(sum(B))}$, 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 . . .)

c) To find the number of allowed sequences that contain the symbol $i$ in the seventh position, we proceed as follows. We are interested in allowed sequernces that have the form

$$x_1, x_2, x_3, x_4, x_5, x_6, i, x_8, x_9, x_{10}.$$
Such sequences consist of an allowed sequence of length 7, that ends in $i$,

$$x_1, x_2, x_3, x_4, x_5, x_6, i,$$

followed by an allowed sequence of length 4 that starts with $i$,

$$i, x_8, x_9, x_{10}.$$  

(Well, technically, we only write down $i$ once when we concatenate the two sequences.) Conversely, if we have any allowed sequence of length 7 that ends in $i$, and any allowed sequence of length 4 that begins with $i$, we can put them together to get an allowed sequence of length 10 that has $i$ in its seventh position. It follows that the total number of sequences of length 10 with $i$ in the seventh position is the product of the number of sequences of length 7 that end in $i$, and the number of sequences of length 4 that begin with $i$. Whew. First let’s find the total number of sequences of length 7 that end in symbol $i$. By the interpretation given above, this is equal to the sum of the entries in the $i^{th}$ row of $A^6$. In a similar way we find that the total number of allowed sequences of length 4 that start with $i$ is given by the sum of the $i^{th}$ column of $A^3$. Carrying out the required computations, we find the numbers of sequences of length 10 with $i$ in the seventh place are, for $i = 1, \ldots , 5$,

$$288, 448, 144, 14, 185.$$  

The most common symbol is $i = 2$. It occurs 448 times out the total of 1079 (which does agree with our previous calculations, which is always good ... ) Some simple matlab code that carries out the required computations is:

```matlab
A = [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]; A_1 = A^6; seq_1 = sum(A_1'); A_2 = A^3; seq_2 = sum(A_2); seq = seq_1 .* seq_2 [occurences, most_common] = max(seq); most_common tot_seq = sum(seq);
```

We should mention a common error. A number of people asserted that to find the most common symbol in the seventh position of a string that has length 10, we only need to find the most common symbol among strings of length 7. In this particular example, it is true that the symbol $i = 2$ is also the most common seventh symbol among all strings of length 7. But in general this can be false. The analysis above shows that you are maximizing a product of two numbers over $i$; the abbreviated method just maximizes the first number. Another error was to try to make the problem into a Markov chain problem, by normalizing the transitions to make them into probabilities, and then finding the symbol with maximum likelihood in the seventh position. This is not only wrong in general, it often led to the wrong answer as well (depending on how the transitions were normalized).
6. Some standard time-series models. A time series is just a discrete-time signal, \( i.e., \) a function from \( \mathbb{Z}_+ \) 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

\[
x(k+1) = Ax(k) + Bu(k)
\]

\[
y(k) = Cx(k) + Du(k)
\]

- 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
\[ \begin{bmatrix} 0 \\ u(k - 1) \\ \vdots \\ u(k + 1 - r) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} u(k - 1) \\ u(k - 2) \\ \vdots \\ u(k - r) \end{bmatrix} \]

so
\[ x(k + 1) = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(k) . \]

\( y(k) \) 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
\[ y(k) = \begin{bmatrix} \overbrace{a_1 & a_2 & \cdots & a_r}^{C} \end{bmatrix} x(k) + \begin{bmatrix} a_0 \end{bmatrix} u(k) . \]

(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
\[
\begin{bmatrix}
0 \\
y(k-1) \\
\vdots \\
y(k+1-p)
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & 0 & \cdots & 0 \\
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{bmatrix}
\begin{bmatrix}
y(k-1) \\
y(k-2) \\
\vdots \\
y(k-p)
\end{bmatrix}
\]
and
\[
\begin{bmatrix}
y(k) \\
0 \\
\vdots \\
0
\end{bmatrix}
= \begin{bmatrix}
b_1 & b_2 & \cdots & b_p \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
y(k-1) \\
y(k-2) \\
\vdots \\
y(k-p)
\end{bmatrix}
+ \begin{bmatrix}
0 \\
u(k)
\end{bmatrix}.
\]
Thus
\[
x(k+1) = \begin{bmatrix}
0 & 0 & 0 & \cdots & 0 \\
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{bmatrix}
x(k) + \begin{bmatrix}
b_1 & b_2 & \cdots & b_p \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix}x(k) + \begin{bmatrix}
1 \\
0 \\
\vdots \\
0
\end{bmatrix}u(k)
\]
or
\[
x(k+1) = \begin{bmatrix}
b_1 & b_2 & b_3 & \cdots & b_p \\
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{bmatrix}A
x(k) + \begin{bmatrix}
1 \\
0 \\
\vdots \\
0
\end{bmatrix}B
\]
and
\[
y(k) = \begin{bmatrix}
b_1 & b_2 & \cdots & b_p \\
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{bmatrix}C
x(k) + \begin{bmatrix}
1 \\
0 \\
\vdots \\
0
\end{bmatrix}D
u(k).
\]

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

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

\[
y(k) = \begin{bmatrix}
y(k-1) \\
y(k-2) \\
\vdots \\
y(k-p)
\end{bmatrix}
\]
and therefore

\[ x(k + 1) = \begin{bmatrix}
0 \\
u(k + 1 - r) \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix} + \begin{bmatrix}
u(k) \\
0 \\
\vdots \\
0
\end{bmatrix} + \begin{bmatrix}
y(k) \\
y(k - 1) \\
y(k - 2) \\
\vdots \\
y(k + 1 - p)
\end{bmatrix}.\]

For similar reasons to the previous parts

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

and

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

Thus

\[
x(k + 1) = \begin{bmatrix}
0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
1 & 0 & 0 & \cdots & 0 & \vdots & \vdots \\
0 & 1 & 0 & \cdots & 0 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0
\end{bmatrix} x(k) + \begin{bmatrix}
1 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix} u(k)
\]

\[
0 & \cdots & 0 & 0 & \cdots & 1 & 0
\end{bmatrix}
\]

18
and
\[ y(k) = \begin{bmatrix} a_1 & a_2 & \cdots & a_r & b_1 & b_2 & \cdots & b_p \end{bmatrix} x(k) + a_0 \ u(k). \]

(Note: it is possible to give state-space models with the state dimension smaller than the ones given here. But our selection of state here makes writing the equations easier.)