# Introduction to Markov Systems

## TRANSITION MATRICES.

Write the transition matrices, state space, and starting state vectors for the following problems. Draw the transition graph. (Write state vectors as columns.)

(1) Four rooms are arranged in a row with doors between adjacent rooms: Room 1 = Room 2 = Room 3 = Room 4
100 people begin in Room 1, and every minute they randomly move to an adjacent room.

[Bonus: Consider this problem with more and with less rooms. Something bad happens with only two rooms.]

(2) There are 60 students equally distributed among three rooms at a party. Every 10 minutes $1/5$ of the students in the dance room go to get drinks and $1/10$ go outside to smoke. After 10 minutes $3/5$ of the smokers keep smoking and the rest all go dance. Also $2/5$ of the students getting drinks will walk outside to smoke and the rest will go dance.

(3) If the cafeteria serves meat today, then tomorrow it will serve chicken $1/3$ of the time and beans $1/6$ of the time. If there is chicken today, then tomorrow is meat $1/3$ of the time and beans $1/6$ of the time. If there is beans today, then tomorrow is meat $2/3$ of the time and chicken $1/3$ of the time. Today there is beans.

(4) In Cyprus 20% of rainy days are followed by a rainy day, while 40% of rainy days are followed by cloudy days. 30% of cloudy days are followed by rainy days and 30% are followed by sunny days. Also 10% of sunny days are followed by rainy days and 20% by cloudy days. Today it is raining.

(5) If the Turkish Lira goes up this week, then next week it will go up again with probability 0.1, but go down with probability 0.6. If the Turkish lira goes down this week, then next week it will go down again with probability 0.5, but go up with probability 0.2. If the Turkish lira remains steady this week, then with probability 0.2 it will go down next week, and with probability 0.05 it will go up. This week the lira held steady.

(6) In a gambling game you begin with 3 TL. Every turn you roll a 6-sided die. If the number is even, you win that much money; if the number is odd you lose that much money. If your roll puts you at (or below) 0 TL, you lose all your money and stop playing. If the roll puts you at (or above) 6 TL then you win 6 TL and stop playing.

(7) [The following problem is for advanced students. For this problem you must allow the state space to grow over time.]

A child plays 2 m from the top of a cliff. Every minute the child randomly moves either 1 m closer to the cliff or 1 m further from the cliff. If the child falls off the cliff, then he stops playing.
(This is how it feels to be a parent, all the time.)

## STABLE STATES.

For the transition matrices given below, draw the transition graph and then find the long term probability.
(State vectors are written as columns.)

(1) $\begin{bmatrix} 1/3 & 2/3 \\ 2/3 & 1/3 \end{bmatrix}$ (2) $\begin{bmatrix} 4/5 & 2/5 \\ 1/5 & 3/5 \end{bmatrix}$ (3) $\begin{bmatrix} 5/6 & 2/3 \\ 1/6 & 1/3 \end{bmatrix}$ (4) $\begin{bmatrix} 1 & 1/5 \\ 0 & 4/5 \end{bmatrix}$ (5) $\begin{bmatrix} 0 & 1/5 \\ 1 & 4/5 \end{bmatrix}$

(6) $\begin{bmatrix} 0 & 1/2 & 0 \\ 1 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$ (7) $\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{bmatrix}$ (8) $\begin{bmatrix} 1/3 & 1/3 & 0 \\ 2/3 & 0 & 1/3 \\ 0 & 2/3 & 2/3 \end{bmatrix}$ (9) $\begin{bmatrix} 1/3 & 0 & 2/3 \\ 1/3 & 1/3 & 0 \\ 1/3 & 2/3 & 1/3 \end{bmatrix}$ (10) $\begin{bmatrix} 1/5 & 1/5 & 1/5 \\ 2/5 & 3/5 & 2/5 \\ 2/5 & 1/5 & 2/5 \end{bmatrix}$

- Recall that we enter matrices into MatLab one row at a time with rows either separated by semicolon or written on separate lines. Similar to elongation matrices, it is often easier to enter transition matrices by column rather than by row; since each column records transitions from a single state, which is how transition data is usually organized. We can then transpose using `'` to change rows to columns.

  For example, consider the transition matrix where state 1 changes to state 2 with probability $1/5$ and to state 3 with probability $3/5$; while state 2 changes to state 1 with probability $1/3$ and to state 3 with probability $1/6$; and state 3 changes to states 1 and 2 each with probability $1/2$.

```
>> T = [1/5  1/5  3/5          %  P(1=>2) = 1/5  and  P(1=>3) = 3/5
        1/3  1/2  1/6          %  P(2=>1) = 1/3  and  P(2=>3) = 1/6
        1/2  1/2   0 ]'        %  P(3=>1) = 1/2  and  P(3=>2) = 1/2
```

  Since we used the transpose `'`, the first **row** that we entered is the first **column** of `T`. The benefit of entering the transition matrix in this way is that it allows you to type the matrix as you read the problem.

- Once the transition matrix is entered, we can calculate states by multiplying. Suppose we begin with 100 people in state 1.

```
>> v0 = [ 100   0   0 ]'       %  state at time 0
>> v1 = T * v0                 %  state at time 1
>> v2 = T * v1                 %  state at time 2
>> v3 = T * v2                 %  state at time 3
```

  This code would give `v0` for the state at $t = 0$, `v1` for $t = 1$, `v2` for $t = 2$, etc. If we want a state much further along we can type

```
>> T * ans                     %  next state
>> T * ans                     %  next state
>> T * ans                     %  next state
```

  using the <up-arrow> on the keyboard to repeat the last command, quickly multiplying by `T` over and over again. If we want a state much further along, say $t = 100$, then this will be tedious. Instead we can multiply by a **power** of `T`!

```
>> v100 = T^100 * v0           %  state at time 100
```

  This is the same as multiplying the state at $t = 0$ by the transition matrix 100 times, but much less boring.

- One way to find the stable state for a Markov matrix is to pick a random initial state and then multiply by the matrix many times. For example

```
>> T = [1/10  3/10  6/10     % P(1=>1) = 1/10  P(1=>2) = 3/10  P(1=>3) = 6/10
        2/10  7/10  1/10     % P(2=>1) = 2/10  P(2=>2) = 7/10  P(2=>3) = 1/10
        3/10  4/10  3/10 ]'  % P(3=>1) = 3/10  P(3=>2) = 4/10  P(3=>3) = 3/10
>> T^50 * [ 0 0 100 ]'      % state after 50 transitions
```

  would compute the state at $t = 50$ for starting with 100 all in state 3. Note that you should get the same answer if you start with 100 in state 1 or 2, or 100 mixed up among the three states, since by the 50th iteration, everything should have stabilized.

```
>> T^50 * [ 100 0 0 ]'         %  each of these
>> T^50 * [ 0 100 0 ]'         %  should give
>> T^50 * [ 30 30 40 ]'        %  the same answer
```

- A slightly better way to find stable states is to use MatLab's `eig` command to give all of the eigenvalues and eigenvectors of a matrix. The stable state corresponds to the eigenvector with eigenvalue 1. The format for the `eig` command is

```
>> [V, D] = eig(T)             %  get eigenvalues and eigenvectors of T
```

  The resulting matrix `V` is the matrix of whose columns are the eigenvectors of `T` and the matrix `D` will have the corresponding eigenvalues on its diagonal. We will discuss this more in the next problem sheet.