MECH 6326 - Optimal Control and Dynamic Programming
Homework 2

Jonas Wagner
jonas.wagner@utdallas.edu

2023, March 3rd

Contents

Markov Chains

Problem 1 Epidemic model simulation

Consider an epidemic model where each individual can be in one of four states: susceptible (S), infected (I), deceased (D), or protected (or immune) (P). An individual can transition from S to I (they become infected), from I to D (they die), from I to S (they recover, but without immunity), or from I to P (they recover, and now have immunity). The states D and P are absorbing: once an individual is in either of these states, they never leave. The transition probabilities for each individual are, with infection states ordered (S,I,D,P),

$$\begin{bmatrix} 0.6 & 0.4 & 0 & 0 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

with an infected neighbor

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

with no infected neighbors.

PROBLEM:

A typical individual has four neighbors: one above, one below, one to the left and one to the right; however, individual on the boundary may be missing one or more of these neighbors. Use Monte Carlo simulation to estimate the expected number of individuals dead at $T = 50$, for a population of 100 individuals on a 10 x 10 grid, starting from the initial distributions:

- Two individuals are infected, one at the (2, 2) location, and the other at the (9, 9) location. The rest of the population is susceptible.
- Two individuals are infected as above, but there are also eight protected individuals, at the following locations:

  $$(2,3),(3,2),(3,7),(4,6),(6,4),(7,3),(8,9),(9,8).$$

  (These individuals could be immunized, for example.) All others are susceptible.

SOLUTION:

Let each individual be considered a node in the fully neighbor connected $n \times n$ square configuration. For each node (i, j), let the state be defined as $\mathcal{X}_{i,j} = \{S, I, D, P\}$. Let The two transition matrices for a single node will be defined as

$$P_1 = \begin{bmatrix} 0.6 & 0.4 & 0 & 0 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A node-based interaction can be setup where $P_{i,j} = P_1$ if a neighbor is infected and $P_{i,j} = P_2$ if no neighbor is infected. The entire system can then be structured as a layered markov chain like structure in $\mathbb{R}^{n \times n \times 4}$. This is written as a neighbor dependent update equation in a function and then ran as many monte calrlo simulations in MATLAB. The number dead for each simulation becomes the sum of all in the dead state at $N = 50$.

1a)

$E[\sum_{i,j} x_{i,j}^{(3)}] \approx 77.36$. (well technically it'd be either 77 or 78...)

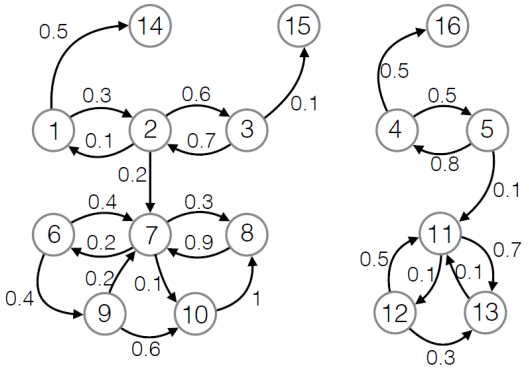
1b)

$E[\sum_{i,j} x_{i,j}^{(3)}] \approx 69.02$. (well technically it'd be either 69 or 70...)

See attached MATLAB scripts.

Problem 2 Computing limiting transition matrix power and steady state distributions.

Consider the following Markov Chain transition graph (self-loops are not shown and can be determined from outgoing edges):



2a)

PROBLEM:

Identify and classify the communicating classes as transient, ergodic, absorbing, or periodic.

SOLUTION:

- Transient: (1,2,3), (4,5)
- Ergodic: ... Probably a bunch... I forgot how to classify ergodic vs periodic
- Absorbing: 14, 15, 16
- Periodic: (11, 12, 13), (6,7,8,9,10)

2b)

PROBLEM:

Do transition matrix powers for this chain converge? If so, compute the limiting value $L = \lim_{t\to\infty} P^t$.

SOLUTION:

The transition matrix equivalent of the markov chain is defined as:

$$P = \begin{bmatrix}
0.2 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.3 & 0.1 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.6 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.5 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.2 & 0 & 0 & 0 & 0.4 & 0.4 & 0.9 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.2 & {} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 & 0.9 & 0 & 0 & 0 \\
0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}^T$$

So this matrix powers are not divergent, however the matrix is both singular and contains multiple eigenvalues with $|\lambda| = 1$ so this leads to it not converging; however, it does appear to reach some form of steady-state experimentally/through estimation, thus the approximate solution from matlab is as follows:

$$L \approx P^{500} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.027 & 0.071 & 0.062 & 0 & 0 & 0.120 & 0.120 & 0.120 & 0.120 & 0.120 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.107 & 0.285 & 0.250 & 0 & 0 & 0.481 & 0.481 & 0.481 & 0.481 & 0.481 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.056 & 0.151 & 0.132 & 0 & 0 & 0.254 & 0.254 & 0.254 & 0.254 & 0.254 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.013 & 0.036 & 0.031 & 0 & 0 & 0.060 & 0.060 & 0.060 & 0.060 & 0.060 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.019 & 0.050 & 0.044 & 0 & 0 & 0.084 & 0.084 & 0.084 & 0.084 & 0.084 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0118 & 0.0235 & 0 & 0 & 0 & 0 & 0 & 0.118 & 0.118 & 0.118 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0015 & 0.0029 & 0 & 0 & 0 & 0 & 0 & 0.015 & 0.015 & 0.015 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0868 & 0.1735 & 0 & 0 & 0 & 0 & 0 & 0.868 & 0.868 & 0.868 & 0 & 0 & 0 \\
0.694 & 0.185 & 0.162 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0.083 & 0.222 & 0.319 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0.9 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

2c)

PROBLEM:

Do state distributions converge to a unique value for any initial distribution? Compute steady-state distributions for the following initial distribution:

- $d_0 = [1, 0, \ldots, 0]$
- $d_0 = [0, 0, 0, 1, 0, \ldots, 0]$
- $d_0 = [0, 0.5, 0, 0, 0.5, 0, \ldots, 0]$

SOLUTION:

- $(d_0^T L)^T = \begin{bmatrix} .2 \\ 0.3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \end{bmatrix}$
- $(d_0^T L)^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \end{bmatrix}$
- $(d_0^T L)^T = \begin{bmatrix} .05 \\ 0.05 \\ 0.3 \\ 0.4 \\ 0.05 \\ 0 \\ 0.1 \\ 0 \\ 0 \\ 0 \\ 0.05 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Problem 3 Absorbing Markov Chains

PROBLEM:

You and 7 of your friends find a $100 bill on the floor. No one has change to allow splitting it evenly, so you decide to play a game of chance to divide the money probabilistically. You all sit around a table. The game is played in turns. Each turn, one of three things happens with equal probability: the bill can move one position to the left, one position to the right, or the game ends and the person with the bill in front of them wins the game. You are the oldest in the group, so the bill starts in front of you. What are the chances you win the money?

SOLUTION:

A markov chain can describe the situation.
Let $\mathcal{X} \in \{1, \ldots, n\}$ where $n = 7$. Let $d_k \in \mathbb{R}^n$ represent the probability that each has the $100 bill. For index i, $i+1$ is to the right and $i-1$ is to the left, and for $i = 0$ the left is $j = n$ and for $i = n$ to the right is $j = 0$.
First define three transition matrices to describe each of the operations. For the cases, $P_w = p_w I_n$, $P_r = p_r[p_{i,i+1} = 1, p_{0,n} = 1]$, and $P_l = p_l[p_{i,i-1} = 1, p_{n,0} = 1]$.
When P_w is enacted, the state enters an absorbing state, thus this can be described in many ways, but one involves instead having an augmented the states to be $d_k \in \mathbb{R}^{2n}$ and the states $\mathcal{X}^{\infty}\{n+1, \ldots, 2n\}$ are absorbing states relating to winning the event.
For the augmented transition matrix the absorbing states is updating as $\mathbf{I}_n$, and the resultant transition matrix for the entire markov chain is defined by:

$$P = \begin{bmatrix} P_{l,r} & P_w \\ \mathbf{0}_n & \mathbf{I}_n \end{bmatrix}$$

and since $p_w = p_r = p_l = \frac{1}{3}$,

$$P_{l,r} = (P_r + P_l) = \frac{1}{3}\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

and $P_w = \frac{1}{3}\mathbf{I}_n$.
For $n = 7$, this results in:

$$P_{l,r} = \frac{1}{3}\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and $\mathbf{I}_7$.
The result for the probability at steady-state is given by $L = \lim_{k\to\infty} d_0^T P^k$, and since the absorbing states are $(n+1), \cdots, 2n$, the probability should end with a vector $d_\infty = \begin{bmatrix} \mathbf{0}_n \\ p_i^{(\infty)} \forall_{i=1,\ldots,n} \end{bmatrix}$.

Out of simplicity, the solution can be approximated as $L \approx P^{100}$ and then

$$d_0^T L = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.4483 \\ 0.1724 \\ 0.0690 \\ 0.0345 \\ 0.0345 \\ 0.0690 \\ 0.1724 \end{bmatrix}$$

Therefore for $n = 7$, the approximation is $P(p_1 = 1) \approx 44.83\%$.

Problem 4 Hitting Times

Consider a Markov Chain with states $\mathcal{X} = \{1, 2, 3, 4\}$. The transition matrix is

$$P = \begin{bmatrix} 0 & 0 & 0.4 & 0.6 \\ 0.3 & 0 & 0.3 & 0.4 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.7 & 0.3 & 0 \end{bmatrix}$$

Each part below defines a type of hitting time τ.

PROBLEM:

For each part, plot the distribution of τ given that the chain starts in state $x_0 = 1$, and also determine $P(\tau = 10 \ : \ x_0 = 1)$.

SOLUTION:

One method of obtaining τ is by augmenting the system with an absorbing state that when reaching a state in $\mathcal{E}$ will be sent to instead of continuing and then

$$P(\tau = k) = \sum_{i \in \mathcal{E}} \sum_{j=1}^{n} p_{i,j}$$

For each section this can be done using this simple process.

4a)

PROBLEM:

The hitting time τ_E, where $E = \{4\}$.

SOLUTION:

For calculating τ_E, the augmented transition matrix becomes

$$P = \begin{bmatrix} 0 & 0 & 0.4 & 0.6 & 0 \\ 0.3 & 0 & 0.3 & 0.4 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The result for $\tau = 10$ is $P(\tau = 10 \ : \ x_0 = 1) = 0.1\%$ along with the distribution in Figure 1.

4b)

PROBLEM:

The hitting time is the smallest time such that $x_t = 4$ and $x_s = 2$ for some $s < t$, (i.e.) the first time state 4 is visited *after* state 2 has been visited. *Hint:* Augment the chain with a copy of each state that encodes whether or not state 2 has been visited yet.

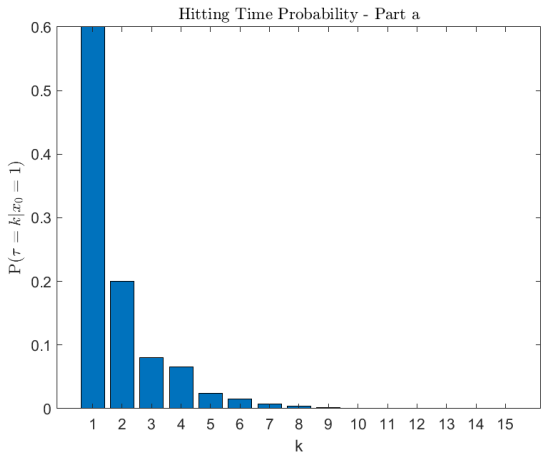


Figure 1: Distribution for Problem 4a

SOLUTION:

This problem can be approched in a similar matter; however first the augmented states will be setup to account for when $x_s = 2$. This would make

$$P = \begin{bmatrix} 0 & 0 & 0.4 & 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0 & 0.3 & 0.4 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0.3 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The result for $\tau = 10$ is $P(\tau = 10 \ : \ x_0 = 1) = 6.62\%$ along with the distribution in Figure 2.

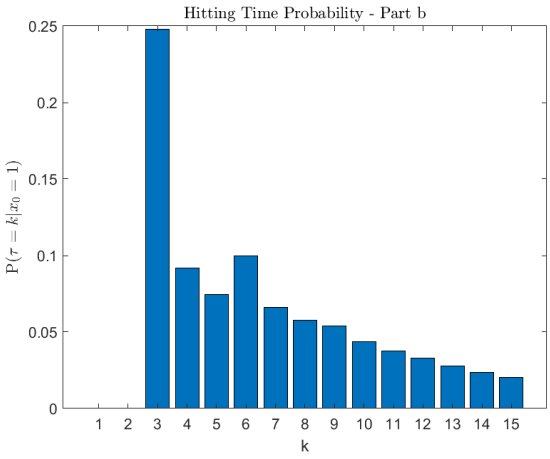


Figure 2: Distribution for Problem 4b

MECH 6326 - HW 3

Author: Jonas Wagner

Date: 2023-04-03

```
clear
close all
```

Problem 1

```
clear
% Matrix Setup
P_1 = eye(4);
P_1(1,:) = [0.6, 0.4, 0, 0];
P_1(2,:) = [0.2, 0.6, 0.1, 0]
```

```
P_1 = 4×4
    0.6000    0.4000         0         0
    0.2000    0.6000    0.1000         0
         0         0    1.0000         0
         0         0         0    1.0000
```

```
P_2 = eye(4);
P_2(2,:) = [0.2, 0.6, 0.1, 0]
```

```
P_2 = 4×4
    1.0000         0         0         0
    0.2000    0.6000    0.1000         0
         0         0    1.0000         0
         0         0         0    1.0000
```

Part 1

```
rng(1)
% State Space
n = 10;
X_0 = zeros(n,n,4);
X_0(:,:,1) = 1;
I_0 = [
    2,2;
    9,9
];
P_0 = [];
for i = 1:size(I_0,1)
    X_0(I_0(i,1),(I_0(i,2)),1) = 0;
    X_0(I_0(i,1),(I_0(i,2)),2) = 1;
end
for i = 1:size(P_0,1)
    X_0(P_0(i,1),P_0(i,2),1) = 0;
    X_0(P_0(i,1),P_0(i,2),2) = 1;
end
```

```
% Run Simulation
nSims = 50;
N = 50;

for i = 1:nSims
    X{i} = X_0;
    for k = 1:N
        X{i} = MECH6326_HW3_pblm1_evolution(X{i},P_1,P_2);
    end
    dead(i) = sum(sum(X{i}(:,:,3)));
end
avgDead = mean(dead)
```

```
avgDead = 77.3600
```

```
maxDead = max(dead)
```

```
maxDead = 88
```

Part2

```
rng(1)
% State Space
n = 10;
X_0 = zeros(n,n,4);
X_0(:,:,1) = 1;
I_0 = [
    2,2;
    9,9
];
P_0 = [
    2,3;
    3,2;
    3,7;
    4,6;
    6,4;
    7,3;
    8,9;
    9,8
];
for i = 1:size(I_0,1)
    X_0(I_0(i,1),(I_0(i,2)),1) = 0;
    X_0(I_0(i,1),(I_0(i,2)),2) = 1;
end
for i = 1:size(P_0,1)
    X_0(P_0(i,1),P_0(i,2),1) = 0;
    X_0(P_0(i,1),P_0(i,2),4) = 1;
end

% Run Simulation
nSims = 50;
```

```
N = 50;

for i = 1:nSims
    X{i} = X_0;
    for k = 1:N
        X{i} = MECH6326_HW3_pblm1_evolution(X{i},P_1,P_2);
    end
    dead(i) = sum(sum(X{i}(:,:,3)));
end
avgDead = mean(dead)
```

```
avgDead = 69.0200
```

```
maxDead = max(dead)
```

```
maxDead = 76
```

Problem 2

```
clear
P = [
0.2    0.1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0.3    0.1    0.7    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0.6    0.2    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0.8    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0.5    0.1    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0.2    0.2    0    0    0    0    0    0    0    0    0
0    0.2    0    0    0    0.4    0.4    0.9    0.2    0    0    0    0    0    0    0
0    0    0    0    0    0    0.3    0.1    0    1    0    0    0    0    0    0
0    0    0    0    0    0.4    0    0    0.2    0    0    0    0    0    0    0
0    0    0    0    0    0    0.1    0    0.6    0    0    0    0    0    0    0
0    0    0    0    0.1    0    0    0    0    0    0.2    0.5    0.1    0    0    0
0    0    0    0    0    0    0    0    0    0    0.1    0.2    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0.7    0.3    0.9    0    0    0
0.5    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
0    0    0.1    0    0    0    0    0    0    0    0    0    0    0    1    0
0    0    0    0.5    0    0    0    0    0    0    0    0    0    0    0    1
]';

eigP = eig(P)
```

```
eigP = 16×1 complex
  -0.5227 + 0.0000i
   0.2000 + 0.0000i
   0.8227 + 0.0000i
  -0.5844 + 0.0000i
   0.6844 + 0.0000i
   1.0000 + 0.0000i
  -0.3433 + 0.3159i
  -0.3433 - 0.3159i
   0.2933 + 0.3592i
   0.2933 - 0.3592i
      ⋮
```

```
L = round(P^500,7)
```

```
L = 16×16
         0         0         0         0         0    0.0267    0.1070    0.0564 ···
         0         0         0         0         0    0.0713    0.2852    0.1505
         0         0         0         0         0    0.0624    0.2496    0.1317
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0    0.1203    0.4813    0.2540
         0         0         0         0         0    0.1203    0.4813    0.2540
         0         0         0         0         0    0.1203    0.4813    0.2540
         0         0         0         0         0    0.1203    0.4813    0.2540
         0         0         0         0         0    0.1203    0.4813    0.2540
     ⋮
```

```
% Part 3
n = size(P,1);
d_0(n,1) = 0; d_0(1,1) = 1;
d_infty = (d_0'*P)'
```

```
d_infty = 16×1
    0.2000
    0.3000
         0
         0
         0
         0
         0
         0
         0
         0
     ⋮
```

```
d_0 = zeros(n,1);
d_0(4,1) = 1;
d_infty = (d_0'*P)'
```

```
d_infty = 16×1
         0
         0
         0
         0
    0.5000
         0
         0
         0
         0
         0
     ⋮
```

```
d_0 = zeros(n,1);
d_0(2,1) = 0.5; d_0(5,1) = 0.5;
d_infty = (d_0'*P)'
```

```
d_infty = 16×1
    0.0500
    0.0500
    0.3000
```

```
    0.4000
    0.0500
         0
    0.1000
         0
         0
         0
    ⋮
```

Problem 3

```
clear
n = 7;
P_w = (1/3)*eye(7);
P_l = [P_w(2:end,:); P_w(1,:)];
P_r = [P_w(end,:); P_w(1:end-1,:)];
P_lr = (P_l + P_r);
P = [P_lr, P_w; zeros(n), eye(n)]
```

```
P = 14×14
         0    0.3333         0         0         0         0    0.3333    0.3333 ···
    0.3333         0    0.3333         0         0         0         0         0
         0    0.3333         0    0.3333         0         0         0         0
         0         0    0.3333         0    0.3333         0         0         0
         0         0         0    0.3333         0    0.3333         0         0
         0         0         0         0    0.3333         0    0.3333         0
    0.3333         0         0         0         0    0.3333         0         0
         0         0         0         0         0         0         0    1.0000
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
    ⋮
```

```
L = P^100
```

```
L = 14×14
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.4483 ···
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.1724
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0690
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0345
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0345
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0690
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.1724
         0         0         0         0         0         0         0    1.0000
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
    ⋮
```

```
d_0(1)=1; d_0(2*n,1) = 0;
d_infty = round((d_0' * L)',7)
```

```
d_infty = 14×1
         0
         0
         0
```

```
         0
         0
         0
         0
    0.4483
    0.1724
    0.0690
     ⋮
```

```
p_win = d_infty(n+1)
```

```
p_win = 0.4483
```

Problem 4

```
clear

P = [
    0   0   0.4 0.6
    0.3 0   0.3 0.4
    0   0.5 0   0.5
    0   0.7 0.3 0
]
```

```
P = 4×4
         0         0    0.4000    0.6000
    0.3000         0    0.3000    0.4000
         0    0.5000         0    0.5000
         0    0.7000    0.3000         0
```

```
% Part a
n = size(P,2);
E = [4];

P_a = P;
P_a(E,:) = 0;
P_a(E,n+1) = 1;
P_a(n+1,n+1) = 1
```

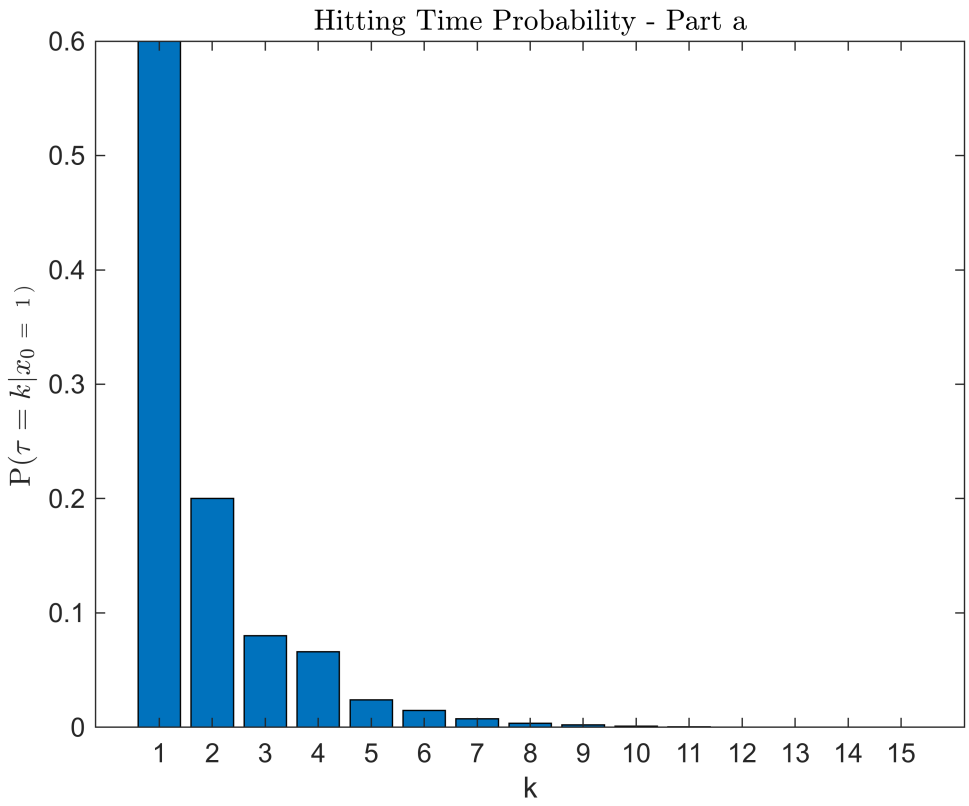
```
P_a = 5×5
         0         0    0.4000    0.6000         0
    0.3000         0    0.3000    0.4000         0
         0    0.5000         0    0.5000         0
         0         0         0         0    1.0000
         0         0         0         0    1.0000
```

```
d_0(n+1,1) = 0;
d_0(1,1) = 1;

tau = 10;
P_tau = P_a^tau;
d_tau = d_0'*P_tau;
p_tau = sum(d_tau(E))
```

```
p_tau = 0.0010
```

```
for tau = 1:15
    P_tau = P_a^tau;
    d_tau = d_0'*P_tau;
    p_tau(tau) = sum(d_tau(E));
end
bar(p_tau)
title("Hitting Time Probability - Part a", 'Interpreter','latex')
xlabel("k")
ylabel("P($\tau = k | x_0 = 1$)",'Interpreter','latex')
saveas(gcf,"figs/pblm4a.png")
```




```
% Part b
E_s = [2];
E_t = [4];
P_11 = P; P_11(E_s,:) = 0;
P_12 = zeros(n); P_12(E_s,:) = P(E_s,:);
P_21 = P; P_21(E_t,:) = 0;
P_b = [P_11, P_12; P_21, zeros(n)];
P_b(n+E_s,2*n+1) = 1;
P_b(2*n+1,2*n+1) = 1
```

```
P_b = 9×9
         0         0    0.4000    0.6000         0         0         0         0 ···
```

```
         0         0         0         0    0.3000         0    0.3000    0.4000
         0    0.5000         0    0.5000         0         0         0         0
         0    0.7000    0.3000         0         0         0         0         0
         0         0    0.4000    0.6000         0         0         0         0
    0.3000         0    0.3000    0.4000         0         0         0         0
         0    0.5000         0    0.5000         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
```

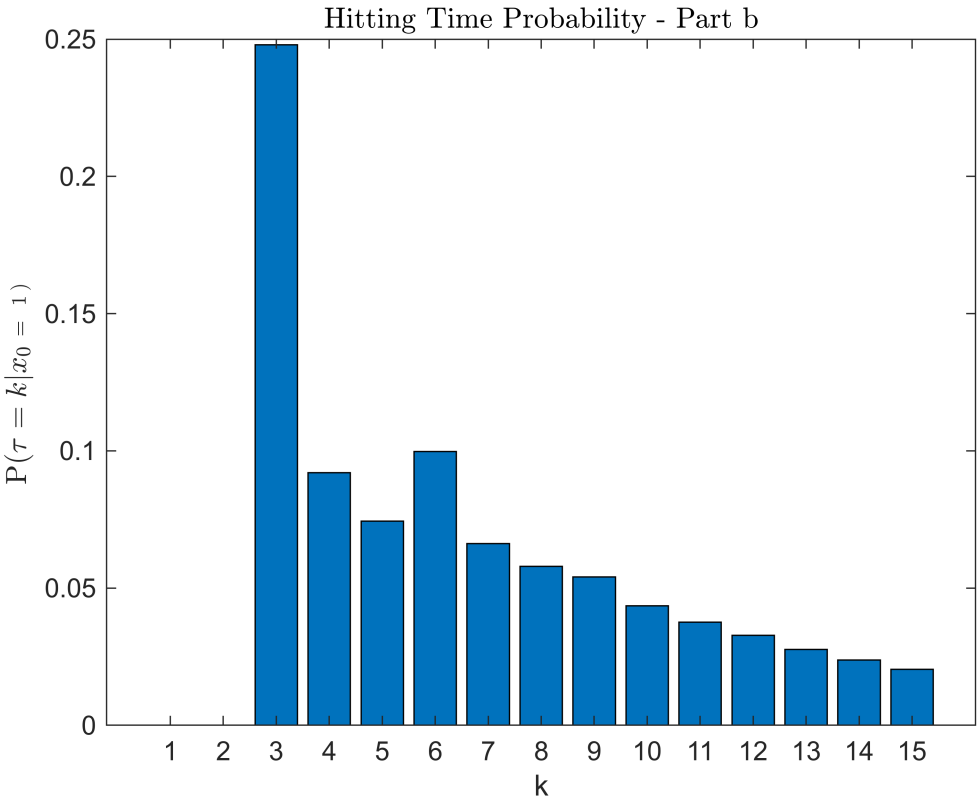
```
d_0 = zeros(2*n+1,1);
d_0(1) = 1
```

```
d_0 = 9×1
     1
     0
     0
     0
     0
     0
     0
     0
     0
```

```
tau = 10;
P_tau = P_b^tau;
d_tau = d_0'*P_tau;
p_tau = sum(d_tau(E))
```

```
p_tau = 0.0662
```

```
for tau = 1:15
    P_tau = P_b^tau;
    d_tau = d_0'*P_tau;
    p_tau(tau) = sum(d_tau(n+E));
end
bar(p_tau)
title("Hitting Time Probability - Part b", 'Interpreter','latex')
xlabel("k")
ylabel("P($\tau = k | x_0 = 1$)",'Interpreter','latex')
saveas(gcf,"figs/pblm4b.png")
```


Hitting Time Probability - Part b
0.25
0.2
0.15
0.1
0.05
0
$\mathrm{P}(\tau = k | x_0 = 1)$
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
k

```
function X = MECH6326_HW3_pblm1_evolution(X,P_1,P_2)
    %MECH6326_HW3_pblm1_evolution
    n = size(X,1);

    % Evolution of each state (dependent on lots of things)
    for i = 1:n
        for j = 1:n
            testvals = combvec(i-1:i+1,j-1:j+1);
            testvals = testvals(:,all([ ...
                all(testvals<=n); ...
                all(testvals>=1); ...
                any(testvals~=[i;j])...
                ]));
            P = P_2;
            for k = 1:size(testvals,2)
                if X(testvals(1,k),testvals(2,k),2) == 1; P = P_1;end
            end
            newState = randsample(4,1,true,reshape(X(i,j,:),[1,4])*P);
            X(i,j,:) = zeros(4,1);
            X(i,j,newState) = 1;
        end
    end
end
```