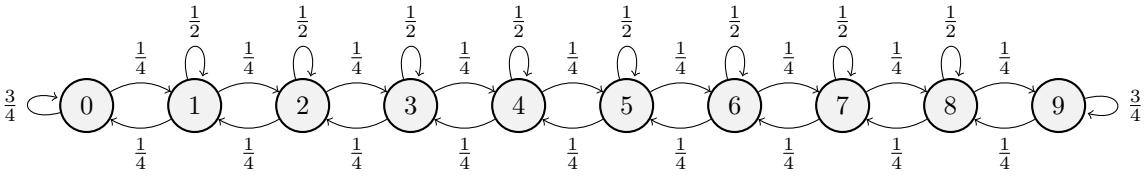
1 Example Sheet 4

1. Draw the state space diagram for this Markov chain.

```
def rw():
        MAX_STATE = 9
        x = 0
        while True:
                yield x
                d = np.random.choice([-1, 0, 1], p=[0.25, 0.5, 0.25])
                x = min(MAX_STATE, max(0, x + d))
```



2. For the Cambridge weather simulator, example 10.2.1 in the lecture notes, show that

$$\mathbb{P}(X_3 = r \mid X_0 = g) = \sum_{x_1, x_2} P_{gx_1} P_{x_1 x_2} P_{x_2 r}$$

$$\begin{aligned} P(X = r \mid X_0 = g) &= P^3_{g,r} \\ &= \sum_{x_1, x_2} P_{gx_1} P_{x_1 x_2} P_{x_2 r} \end{aligned}$$

3. here is the state space diagram for a Markov chain with a state space $\{0, 1, \ldots, N\}$. States 0 and N are absorbing states. at any other state x we jump to $x-1$ with probability α and to $x+1$ with probability β. Assume $\alpha > 0$ and $\beta > 0$ and $\alpha + \beta \leq 1$. Let $h_x = \mathbb{P}(\text{hit } 0 \mid \text{start at } x)$.

   - For $N = 3$, calculate h_x for all $x \in \{0, 1, 2, 3\}$.

$$\begin{aligned} h_0 &= 1 & h_1 &= \beta h_0 + (1-\alpha-\beta)h_1 + \alpha h_2 \\ h_2 &= \beta h_1 + (1-\alpha-\beta)h_2 + \alpha h_3 & h_3 &= 0 \end{aligned}$$

$$\begin{aligned} h_1 &= \beta h_0 + (1-\alpha-\beta)h_1 + \alpha h_2 \\ (\alpha+\beta)h_1 &= \beta + \alpha h_2 \\ h_2 &= \beta h_1 + (1-\alpha-\beta)h_2 + \alpha h_3 \\ (\alpha+\beta)h_2 &= \beta h_1 \\ (\alpha+\beta)h_1 &= \beta + \frac{\alpha\beta}{\alpha+\beta} h_1 \\ h_1 &= \frac{\beta(\alpha+\beta)}{\alpha^2+\alpha\beta+\beta^2} \\ h_2 &= \frac{\alpha\beta}{\alpha^2+\alpha\beta+\beta^2} \end{aligned}$$

$$h_0 = 1 \qquad h_1 = \frac{\beta(\alpha+\beta)}{\alpha^2+\alpha\beta+\beta^2}$$
$$h_2 = \frac{\alpha\beta}{\alpha^2+\alpha\beta+\beta^2} \qquad h_3 = 0$$

- For arbitrary N, give code to compute the vector $[h_0, h_1, \ldots, h_n]$.

```
P = np.zeros((N + 1, N + 1))
P[np.arange(1, N), np.arange(N - 1)] = β
P[np.arange(1, N), np.arange(1, N)] = 1 - α - β
P[np.arange(1, N), np.arange(2, N + 1)] = α
P[0, 0] = 1
P[N, N] = 1
conditions = np.zeros((2, N + 1))
conditions[0, 0] = 1
conditions[-1, -1] = 1
mat = np.concatenate((P - np.eye(N + 1), conditions))
target = np.zeros(N + 3)
target[-2] = 1
np.linalg.lstsq(mat, target, rcond=None)
```

4. Consider a random walk on the vertices of this undirected graph as follows: each timestep we take one of the edges chosen at random, each edge from our current vertex equally likely. Find the stationary distribution.

   The stationary distribution is as follows:

   $$\left(\tfrac{1}{8}, \tfrac{3}{8}, \tfrac{2}{8}, \tfrac{2}{8}\right)$$

   We can calculate this by either applying the standard result for the stationary distribution of a random walk on an undirected graph – π_i is the number of edges i has divided by twice the total number of edges.

   Or we can use the code below:

```
P = np.array([
        [0, 1, 0, 0],
        [1/3, 0, 1/3, 1/3],
        [0, 1/2, 0, 1/2],
        [0, 1/2, 1/2, 0],
])
mat = np.concatenate((P.transpose() - np.eye(4), np.ones((1, 4))))
np.linalg.lstsq(mat,[0, 0, 0, 0, 1], rcond=None)
```

5. Let X_n be a mean-reverting random walk,

   $$X_{n+1} = \mu + \lambda(X_n - \mu) + \mathcal{N}(0, \sigma^2) \quad \text{where} \quad -1 < \lambda < 1$$

   The stationary distribution for this process is a normal distribution. Find its parameters.

   Since the distribution is stationary: $\mathbb{E}(X_{n+1}) = \mathbb{E}(X_n)$

   $$\begin{aligned}
   \mathbb{E}(X_n) &= \mathbb{E}(X_{n+1}) \\
   \mathbb{E}(X_n) &= \mathbb{E}(\mu + \lambda(X_n - \mu) + \mathcal{N}(0, \sigma^2)) \\
   \mathbb{E}(X_n) &= \mu + \lambda(\mathbb{E}(X_n) - \mu) + 0 \\
   (1-\lambda)\mathbb{E}(X_n) &= (1-\lambda)\mu \\
   \mathbb{E}(X_n) &= \mu
   \end{aligned}$$

Therefore if X_n is normally distributed then it has mean μ.

Since the distribution is stationary, $\mathbb{E}(X_n^2) = \mathbb{E}(X_{n+1}^2)$

$$\begin{aligned}
\mathbb{E}(X_n^2) &= \mathbb{E}(X_{n+1}^2) \\
\mathbb{E}(X_n^2) &= \mathbb{E}(\mathcal{N}(\mu + \lambda(X_n - \mu), \sigma^2)^2) \\
\mathbb{E}(X_n^2) &= \mathbb{E}((\mu + \lambda(X_n - \mu))^2 + \sigma^2) \\
\mathbb{E}(X_n^2) &= \mathbb{E}(\mu^2 + 2\lambda\mu X_n - 2\lambda\mu^2 + \lambda^2 X_n^2 - 2\lambda^2 X_n\mu + \lambda^2\mu^2 + \sigma^2) \\
\mathbb{E}(X_n^2) &= \mu^2 + 2\lambda\mu^2 - 2\lambda\mu^2 + \lambda^2\mathbb{E}(X_n^2) - 2\lambda^2\mu^2 + \lambda^2\mu^2 + \sigma^2 \\
\mathbb{E}(X_n^2) &= \frac{(1-\lambda^2)\mu^2 + \sigma^2}{1-\lambda^2} \\
\mathbb{E}(X_n^2) - \mathbb{E}(X_n)^2 &= \frac{(1-\lambda^2)\mu^2 + \sigma^2}{1-\lambda^2} - \mu^2 \\
\mathrm{Var}(X_n) &= \frac{\sigma^2}{1-\lambda^2}
\end{aligned}$$

Therefore if X_n is normally distributed then $X_n \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{1-\lambda^2}\right)$

6. The Internet uses an algorithm called TCP to manage congestion. For every data flow, the sender maintains a congestion window $W_n \in \mathbb{R}$. It keeps roughly W_n packets in flight, thus the transmission rate is $\frac{W_n}{RTT}$ packets per second where RTT is the round trip time between sender and receiver. The sender updates W_n every time it receives an acknowledgement of a packet, by

$$W_{n+1} = \begin{cases} W_n + \frac{1}{W_n} & \text{if the packet didn't experience congestion} \\ \frac{W_n}{2} & \text{if the packet did experience congestion} \end{cases}$$

Suppose that packets experience congestion with probability p_i independently. Write down a drift model for W_n and find the fixed point.

$$\begin{aligned}
\mathbb{E}(W_n) &= \mathbb{E}(W_{n+1}) \\
\mathbb{E}(W_n) &= p_i \cdot \frac{\mathbb{E}(W_n)}{2} + (1-p_i)\left(\mathbb{E}(W_n) + \frac{1}{\mathbb{E}(W_n)}\right) \\
\mathbb{E}(W_n) &= p_i \cdot \frac{\mathbb{E}(W_n)}{2} + \mathbb{E}(W_n) + \frac{1}{\mathbb{E}(W_n)} - p_i \cdot \mathbb{E}(W_n) - p_i \cdot \frac{1}{\mathbb{E}(W_n)} \\
0 &= -\frac{p_i}{2} \cdot \mathbb{E}(W_n) + \frac{1}{\mathbb{E}(W_n)} - p_i \cdot \frac{1}{\mathbb{E}(W_n)} \\
0 &= \frac{p_i}{2} \cdot \mathbb{E}(W_n)^2 - 1 + p_i \\
\mathbb{E}(W_n)^2 &= \frac{2(1-p_i)}{p_i} \\
\mathbb{E}(W_n) &= \sqrt{\frac{2(1-p_i)}{p_i}}
\end{aligned}$$

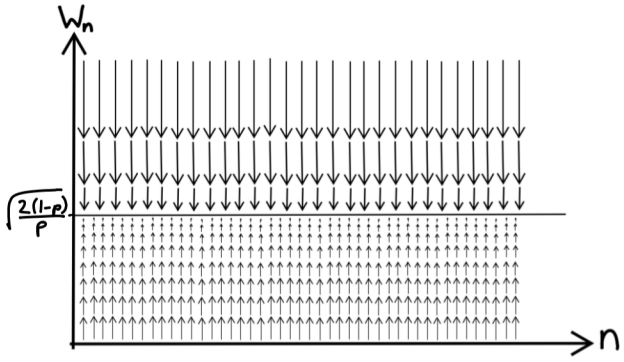


Figure 1: Drift Model for W_n

7. We're given a sequence $(x_0, x_1, \ldots, x_n)$ starting at $x_0 = 0$ and we decide to model it as a mean-reverting random walk as in question 5. Explain how to estimate λ, μ and σ.

   Using question 5:

$$\mathbb{E}(X_n) = \mu \qquad \text{Var}(X_n) = \frac{\sigma^2}{1-\lambda^2}$$

$$\begin{aligned}\text{Var}(X_{n+1} + X_n) &= \text{Var}(\mu + \lambda(X_n - \mu) + \mathcal{N}(0, \sigma^2) + X_n) \\ &= \text{Var}((1-\lambda)\mu + (\lambda+1)X_n + \mathcal{N}(0, \sigma^2)) \\ &= (\lambda+1)\text{Var}(X_n) + \sigma^2 \\ &= \frac{(2-\lambda)\sigma^2}{1-\lambda}\end{aligned}$$

   We now have an unbiased estimator for μ and can use fmin to find a solution for μ and σ for example by using the following code:

```
α = Var(X_n)
β = Var(X_{n+1} − X_n)
def error(x):
        λ, σ = x
        dif_α = α − σ²/(1−λ²)
        dif_β = β − (2−λ)σ²/(1+λ)
        return dif_α ** 2 + dif_β ** 2
opt.fmin(error, [λ_0, σ_0])
```

8. Consider a moving object with noisy location readings. Let X_n be the location at timestep $n \geq 0$ and Y_n the reading. Here's the simulator.

```
def hmm():
        max_state = 9
        x = np.random.randint(0, max_state + 1)
        while True:
                e = np.random.choice([-1, 0, 1])
                y = min(max_state, max(0, x + e))
                yield y
                d = np.random.choice([-1, 0, 1], p=[1/4, 1/2, 1/4])
                x = min(max_state, max(0, x + d))
```

   We'd like to infer the location X_n given readings $y_0, \ldots y_n$.

(a) Give justifications for the following three equations, which give an inductive solution. First the base case,

$$\Pr(x_0 \mid y_0) = c \times \Pr(x_0) \Pr(y_0 \mid x_0)$$

and next two equations for the induction step,

$$\Pr(x_n) = \sum_{x_{n-1}} \Pr(x_{n-1} \mid h) \Pr(x_n \mid x_{n-1})$$
$$\Pr(x_n \mid h, y_n) = c \times \Pr(x_n \mid h) \Pr(y_n \mid x_n)$$

In these two equations, h stands for $(y_0, \ldots, y_{n-1})$ and we'll assume we've already found $\Pr(x_{n-1} \mid h)$.

- $\Pr(x_0 \mid y_0) = c \times \Pr(x_0) \Pr(y_0 \mid x_0)$

  Using Bayes rule:

$$\begin{aligned}\Pr(x_0 \mid y_0) &= \frac{\Pr(x_0) \Pr(y_0 \mid x_0)}{\Pr(y_0)} \\ &= c \times \Pr(x_0) \Pr(y_0 \mid x_0)\end{aligned}$$

- $\Pr(x_n) = \sum_{x_{n-1}} \Pr(x_{n-1} \mid h) \Pr(x_n \mid x_{n-1})$

  Using the law of total probability

$$\begin{aligned}\Pr(x_n \mid h) &= \sum_{x_{n-1}} \Pr(x_n, x_{n-1} \mid h) \\ &= \sum_{x_{n-1}} \Pr(x_{n-1} \mid h) \Pr(x_n \mid x_{n-1}, h)\end{aligned}$$

  Since the next state depends only on the previous state

$$= \sum_{x_{n-1}} \Pr(x_{n-1} \mid h) \Pr(x_n \mid x_{n-1})$$

- $\Pr(x_n \mid h, y_n) = c \times \Pr(x_n \mid h) \Pr(y_n \mid x_n)$

$$\begin{aligned}\Pr(x_n \mid h, y_n) &= \frac{\Pr(y_n \mid x_n, h) \Pr(x_n \mid h)}{\Pr(y_n \mid h)} \\ &= c \times \Pr(y_n \mid x_n, h) \Pr(x_n \mid h)\end{aligned}$$

  Since the observed state depends only on the hidden state

$$= c \times \Pr(y_n \mid x_n) \Pr(x_n \mid h)$$

(b) Give pseudocode for a function that takes as input a list of readings $(y_0, \ldots, y_n)$ and outputs the probability vector

$$[\pi_0, \ldots, \pi_{\text{MAX\_STATE}}] \qquad \pi_x = \mathbb{P}(X_n = x \mid y_0, \ldots, y_n)$$

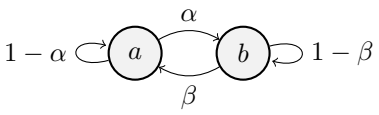
```
def P(x_n, y_n, h):
        π = np.zeros(max_state)
        if y_n = 0:
                π[0] = 2/3
                π[1] = 1/3
        elif y_n = max_state:
                π[-1] = 2/3
                π[-2] = 1/3
        else:
                π[y_n-1:y_n+2] = 1/3
        if h.size == 0:
                return π
        for x_n in states:
                π[x_n] *= np.sum(Pr(x_{n-1}, h) * Pr_transition(x_{n-1}, x_n))
        return π / np.sum(π)

def Pr_transition(x_{n-1}, x_n):
        if x_n == x_{n-1}:
                return 0.5
        elif abs(x_n - x_{n-1}) == 1:
                return 0.25
        else:
                return 0
```

(c) If your code is given the input $(3, 3, 4, 9)$, it should fail with a divide-by-zero error. Give an interpretation of this failure.

The observed dataset is impossible. The probability of the current state being in any state is undefined; normalising undefined data is not mathematically legal to do. So the code throws an exception.

2 Supplementary Questions

10. Here is the state space diagram for a Markov chain. Find a stationary distribution. Is it unique?



With π_x as the x component of the stationary distribution, we can form the equations:

$$\pi_a = (1-\alpha)\pi_a + \beta\pi_b \qquad \pi_b = \alpha\pi_a + (1-\beta)\pi_b \qquad 1 = \pi_a + \pi_b$$

We can solve these to give the following values:

$$\pi_a = \frac{\beta}{\alpha+\beta} \qquad \pi_b = \frac{\alpha}{\alpha+\beta}$$

This stationary distribution is unique.

11. Here is the state space diagram for a Markov chain with state space $\{0, 1, 2, \dots\}$. It is parameterized by α and β, with $0 < \alpha < \beta$ and $\alpha + \beta < 1$. Let $\pi_n = \left(1 - \frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n$, $n \geq 0$. Show that π is a stationary distribution.

Firstly, we must prove that π is a distribution:

$$\begin{aligned}\sum_{n=0}^{\infty}\pi_n &= \sum_{n=0}^{\infty}\left(\frac{\alpha}{\beta}\right)^n \\ &= \left(1-\frac{\alpha}{\beta}\right)\sum_{n=0}^{\infty}\left(\frac{\alpha}{\beta}\right)^n\end{aligned}$$

Using the formula for the sum of a geometric distribution

$$\begin{aligned}&= \left(1-\frac{\alpha}{\beta}\right)\frac{1}{1-\frac{\alpha}{\beta}} \\ &= 1\end{aligned}$$

So π is a valid distribution

To prove π is a *stationary* distribution, we must show that $\mathbf{P}\pi = \pi$ where $\mathbf{P}$ is the transition matrix.

For $n = 0$:

$$\begin{aligned}(\mathbf{P}\pi)_n &= (1-\alpha)\left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n + \beta\left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^{n+1} \\ &= \left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n\left(1-\alpha+\beta\frac{\alpha}{\beta}\right) \\ &= \left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n(1-\alpha+\alpha) \\ &= \left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n \\ &= \pi_n\end{aligned}$$

For $n \neq 0$:

$$\begin{aligned}(\mathbf{P}\pi)_n &= \sum_{i=0}^{\infty}\mathbf{P}_{n,i}\pi_i \\ &= \mathbf{P}_{n,n-1}\pi_{n-1} + \mathbf{P}_{n,n}\pi_n + \mathbf{P}_{n,n+1}\pi_{n+1} \\ &= \alpha\left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^{n-1} + (1-\alpha-\beta)\left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n + \beta\left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^{n+1} \\ &= \left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n(\beta+1-\alpha-\beta+\alpha) \\ &= \left(1-\frac{\alpha}{\beta}\right)\left(\frac{\alpha}{\beta}\right)^n \\ &= \pi_n\end{aligned}$$

12. Let $X_n \in \mathbb{N}$ be te number of infections people on day n of an epidemic and consider the Markov chain model

$$X_{n+1} = X_n + \mathbf{Po}\left(\frac{rX_n}{d}\right) - \mathbf{B}\left(X_n, \frac{1}{d}\right)$$

We would like to compute the probability that, starting from state $X_0 = x$, the epidemic dies out i.e. hits state 0. In order to solve this by computer, we'll cut the state space down to $\{0, \dots N\}$ for some sufficiently large N by amalgamating all the states with $\geq N$ infected and letting the transition from state N back to itself have probability 1.

(a) Give pseudocode to compute the transition matrix. You should give your answer in terms of `binom.pmf` and `poisson.pmf`, the likelihood functions for the two distributions in question.

```
P = np.zeros((N, N))
for i in range(1, N-1):
        for j in range(N):
                prob = stats.poisson.pmf(j - i + np.arange(i+1), r * i / d) * \
                stats.binom.pmf(np.arange(i+1), i, 1 / d)
                P[i, j] = np.sum(prob)
P[0, 0] = 1
P[-1, -1] = 1
P /= np.sum(P, axis=1, keepdims=True)
```

(b) Give pseudocode to compute the probability that the epidemic dies out, starting from any initial state $x \in \{0, \dots, N\}$.

```
condition1 = np.zeros(N)
condition1[0] = 1
condition2 = np.zeros(N)
condition2[-1] = 1
matrix = np.concatenate((P - np.eye(N), [condition1, condition2]))
print(matrix)
target = np.zeros(N + 2)
target[-2] = 1
np.linalg.lstsq(matrix, target, rcond=None)
```

13. Consider a directed acyclic graph representing the web, with one vertex per webpage and an edge $v \to w$ if page v links to page w. Consider a random web surfer who goes from page to page according to the algorithm.

```
d = 0.85
def next_page(v):
        neighbours = list of pages w wuch that v → w
        a = random.choice(['follow_link', 'teleport'], p=[d, 1-d])
        if a == 'follow_link' and len(neighbours) > 0:
                return random.choice(neighbours)
        else:
                V = list of all web pages
                return random.choice(V)
```

This solves a Markov chain. Explain why the chain is irreducible. Show that the stationary distribution π solves

$$\pi_v = \frac{1-d}{|V|} + d \sum_{u:u\to v} \frac{\pi_u}{|\Gamma_u|}$$

where $|V|$ is the total number of web pages in the graph and $|\Gamma_u|$ is the number of outgoing edges from u.

Compute the stationary distribution for this random web surfer model, for the graph in lecture notes example 10.3.1. Repeat with $d = 0.05$. What do you expect as $d \to 0$? What do you expect if $d = 1$?

Irreducible means that there is a path from any state to any other state. Since we can teleport from one state to any other state, we can clearly reach any state from any other state.

The stationary distribution of the markov chain must maintain the invariant that the probability of being in the state v is the same as the probability of being in the state v after a step. Therefore:

$$\pi_v = d \cdot (\mathbf{P}\pi)_v + \frac{1-d}{|V|}$$
$$\pi_v = \frac{1-d}{|V|} + d \sum_{u:u\to v} \frac{\pi_u}{|\Gamma_u|}$$

As required

```
edges = np.array([
    [0, 1, 1, 0, 0, 0],
    [1, 0, 0, 0, 1, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 1, 0, 1, 1],
    [0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 1]
])
n = edges.shape[0]
P = edges / np.sum(edges, axis=1, keepdims=True)
P *= d
P += (1 - d) / n
matrix = np.concatenate((P.transpose() - np.eye(n), [np.ones(n)]))
np.linalg.lstsq(matrix, [0, 0, 0, 0, 0, 0, 1], rcond=None)
```

For $d = 0.85$ this returns:

$$\pi = \begin{pmatrix} 0.047839 \\ 0.080608 \\ 0.083004 \\ 0.132961 \\ 0.085511 \\ 0.570076 \end{pmatrix}$$

For $d = 0.05$ this returns:

$$\pi = \begin{pmatrix} 0.161108 \\ 0.166491 \\ 0.165205 \\ 0.170661 \\ 0.163953 \\ 0.172582 \end{pmatrix}$$

As $d \to 0$, the distribution will tend to $\pi = (0, 0, 0, 0, 0, 1)$. If $d = 1$ then the distribution π is $\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right)$.

14. Consider the random web surfer model for the graph in lecture notes example 10.3.1. Let t_x be the expected time, starting from vector x until the server reaches vertex 5. Derive the equations:

$$t_x = 1 + \sum_y P_{xy} t_y \quad \text{for all x} \neq 5, \quad t_5 = 0$$

and solve with numpy

```
edges = np.array([
    [0, 1, 1, 0, 0, 0],
    [1, 0, 0, 0, 1, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 1, 0, 1, 1],
    [0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 1]
])
n = edges.shape[0]
P = edges / np.sum(edges, axis=1, keepdims=True)
P -= np.eye(n)
P = np.concatenate((P, [[0, 0, 0, 0, 0, 1]]))
np.linalg.lstsq(P, [-1, -1, -1, -1, -1, 0, 0], rcond=None)
```

The result is:

$$\begin{pmatrix} 6.73 \\ 5.27 \\ 6.18 \\ 5.09 \\ 6.09 \\ 0.00 \end{pmatrix}$$

15. The code from question 8(c) can fail with a divide-by-zero error. This is undesirable in production code! One way to fix the problem is ot modify the Markov model to include a "random teleport" – to express the idea "Okay, our interface has gone wrong somewhere; let's allow our location estimate to reset itself". We can achieve this mathematically with the following model: with probability $1-\varepsilon$ generate the next state as per line 9, otherwise pick the next state uniformly from $\{0, 1, \ldots, \text{MAX\_STATE}\}$. Modify your code from question b to reflect this new model with $\varepsilon = 0.01$.

    The only changes necessary are to the function `Pr_transition`:

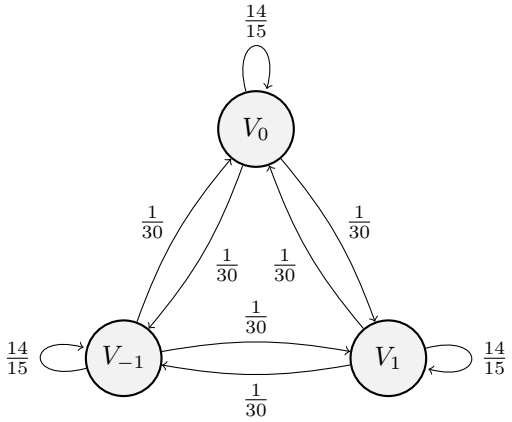
```
def Pr_transition(x_{n-1}, x_n):
        if x_n == x_{n-1}:
                p = 0.5
        elif abs(x_n - x_{n-1}) == 1:
                p = 0.25
        else:
                p = 0
        p *= (1 - ε)
        p += ε / (max_state + 1)
        return p
```

    Alternatively, we could fix the problem by changing the model to express "Okay, this reading is glitchy; let's allow the code to discard an impossible reading". How might you change the Markov model to achieve this?

    We can give all states an infinitesimally small probability δ of emitting a random observation. With δ small enough, an impossible observation becomes an unknown observation while δ will not meaningfully affect the probabilities of known observations. If there is a known probability of a glitchy reading, we may want to have a non-negligible probability.

16. The Markov Model for motion from question 1 is called a *simple random walk (with boundaries)*; it chooses a direction fo travel independently at every timestep. This is not a good model for human movement, since people tend to head in the same direction for a while before changing direction.

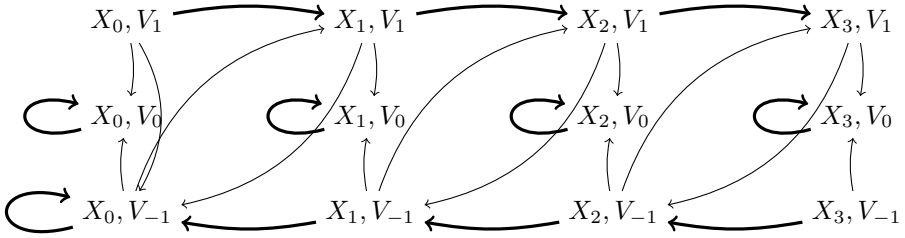
(a) Let $V_n \in \{-1, 0, 1\}$ be a Markov chain: let $V_{n+1} = V_n$ with probability 0.9, and let V_{n+1} be chosen uniformly at random from $\{-1, 0, 1\}$ with probability 0.1. Draw a state space diagram for this Markov chain.



(b) Interpret V_n as the velocity of our moving object at timestep n, and let $X_{n+1} = \max(0, \min(9, X_n + V_n))$.

Draw the state space diagram for (X_n, V_n).

For brevity, normal lines have probability $\frac{1}{30}$ and thick lines have probability $\frac{14}{15}$. I have only drawn the first four X_i due to space constraints.



(c) Give pseudocode to compute the stationary distribution.

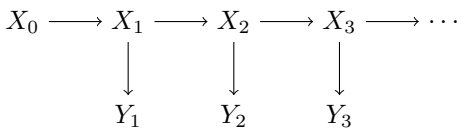
```
probs = np.zeros((30, 30))
#  0-9 is x0v-1 - x9v-1
# 10-19 is x0v-1 - x9v-1
# 20-29 is x0v-1 - x9v-1
probs[np.arange(1, 10), np.arange(9)] = 14/15
probs[0, 0] = 14/15
probs[np.arange(10), np.arange(10, 20)] = 1/30
probs[np.arange(9), np.arange(21, 30)] = 1/30
probs[9, 29] = 1/30

probs[np.arange(10, 20), np.arange(10, 20)] = 14/15
probs[np.arange(11, 20), np.arange(0, 9)] = 1/30
probs[10, 0] = 1/30
probs[np.arange(10, 19), np.arange(21, 30)] = 1/30
probs[19, 29] = 1/30

probs[np.arange(20, 29), np.arange(21, 30)] = 14/15
probs[29, 29] = 14/15
probs[np.arange(20, 30), np.arange(10, 20)] = 1/30
probs[np.arange(21, 30), np.arange(9)] = 1/30
probs[20, 0] = 1/30
```

```
mat = np.concatenate((probs.transpose() - np.eye(30), np.ones((1, 30))))

target = np.zeros(31)
target[30] = 1

np.linalg.lstsq(mat, target, rcond=None)
```

3 2020 Paper 6 Question 7

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2020p6q7.pdf

Consider the probability model



where $(X_0, X_0, \dots)$ is a Markov chain on state space $\{0, 1\}$ with transition probabilities $P_{01} = p$, $P_{10} = q$; and where each Y_i is normally distributed with mean X_i and variance σ^2.

We are given a sequence of obserations $(y_1, y_2, \dots, y_n)$, and we wish to make an inference about the unobserved values $(X_1, X_2, \dots, X_n)$. We will take $0 < p < 1$, $0 < q < 1$ and $\sigma > 0$ to be known and we will assume that X_0 is sampled from the Markov Chain's stationary distribution.

(a) Write out the transition matrix for the Markov Chain $(X_0, X_1, \dots)$. Calculate its stationary distribution.

With $\mathbf{P}$ as the transition matrix for the Matrix Chain:

$$\mathbf{P} = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$

The stationary distribution π satisfies $\mathbf{P}\pi = \pi$. Therefore:

$$\begin{aligned} (1-p)\pi_0 + q\pi_1 &= \pi_0 \\ q\pi_1 - p\pi_0 &= 0 \\ q\pi_1 - p(1-\pi_1) &= 0 \\ (p+q)\pi_1 &= p \\ \pi_1 &= \frac{p}{p+q} \end{aligned} \qquad \begin{aligned} p\pi_0 + (1-q)\pi_1 &= \pi_1 \\ p\pi_0 - q\pi_1 &= 0 \end{aligned} \qquad \begin{aligned} \pi_0 + \pi_1 &= 1 \\ \pi_0 + \pi_1 &= 1 \end{aligned}$$

Therefore:

$$\pi_0 = \frac{q}{p+q} \qquad \pi_1 = \frac{p}{p+q}$$

(b) Writing $\overrightarrow{X}$ for $(X_0, X_1, \dots, X_n)$ and writing $\overrightarrow{Y}$ for $(Y_1, \dots, Y_n)$ and similarly $\overrightarrow{x}$ and $\overrightarrow{y}$, find expressions for:

$$\mathbb{P}(\overrightarrow{X} = \overrightarrow{x}) \text{ and for } \mathbb{P}(\overrightarrow{Y} = \overrightarrow{y} \mid \overrightarrow{X} = \overrightarrow{x})$$

I will use list slicing notation $[:i]$ to mean "the first i values". For example $\overrightarrow{x}[:2]$ means x_0, x_1.

$$\mathbb{P}(\overrightarrow{X} = \overrightarrow{x}) = \mathbb{P}(x_0) \cdot \mathbb{P}(x_1 \mid \overrightarrow{x}[:1]) \cdot \cdots \cdot \mathbb{P}(x_n \mid \overrightarrow{x}[:n])$$

Using Memorylessness:

$$\begin{aligned}\mathbb{P}(\overrightarrow{X} = \overrightarrow{x}) &= \mathbb{P}(x_0) \cdot \mathbb{P}(x_1 \mid x_0) \cdot \cdots \cdot \mathbb{P}(x_n \mid x_{n-1}) \\ &= \pi_{x_0} \cdot \prod_{i=0}^{n-1} \mathbf{P}_{X_i X_{i+1}}\end{aligned}$$

Since the observed value of a Markov Chain depends only on its current state:

$$\begin{aligned}\mathbb{P}(\overrightarrow{Y} = \overrightarrow{y} \mid \overrightarrow{X} = \overrightarrow{x}) &= \prod_{i=1}^{n} \Pr(y_i \mid x_i) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - x_i)^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{2\sigma^2}}\end{aligned}$$

(c) Give pseudocode for a function `rx(n)` that generates a random $\overrightarrow{X}$. Give pseudocode to generate a weighted sample from the posterior distribution of $\overrightarrow{X}$ conditional on the observed data $\overrightarrow{Y} = \overrightarrow{y}$.

```
def rx(n):
    x = np.random.choice(2, 1, π)
    for i in range(n):
        x.append(np.random.choice(2, p=π[x[-1]]))
    return x
```

We can generate a weighted sample by randomly generating samples $\overrightarrow{x}$ using `rx(n)` and sampling using $\mathbb{P}(\overrightarrow{y} \mid \overrightarrow{x})$ as the weight.

```
def P(x, y):
    # work out P(y | x)
    p = [1/√(2πσ) · e^(-(y_i - x_i)^2 / 2σ^2) for x_i, y_i in zip(x, y)]
    prob = 1
    for p_i in p:
        prob *= p_i
    return prob

def weighted_rx(y, samples=1000):
    samples = [rx(len(y)) for _ in range(samples)]
    probs = [P(x, y) for x in samples]
    return samples[np.random.choice(samples + 1, p=probs / sum(probs))]
```

(d) Let $Z = \frac{\sum_{i=1}^{n} X_i}{n}$. Give pseudocode to find a 95% confidence interval for Z, conditional on the observed data $\overrightarrow{Y} = \overrightarrow{y}$.

```
lo = 0.025
hi = 0.975
```

```
xs = [weighted_rx(y) for _ in range(1000)]
zs = np.mean(xs[:, 1:], axis=1)
np.quantile(zs, [lo, hi])
```

4 2022 Paper 6 Question 8

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2022p6q8.pdf

Let $(E_0, E_1, \dots)$ be a Markov Chain generated by

$$E_{t+1} = \lambda E_i + \mathcal{N}(0, (1-\lambda^2)\sigma^2)$$

where $0 \leq \lambda < 1$

(a) What is meant by a stationary distribution?

A stationary distribution is a probability vector π such that the probability of being in state i at time t is equal to π_i for all t.

(b) What is meant by "memorylessness"? Give an expression for the log likelihood of a sequence of values $(e_1, \dots, e_n)$ given the value for E_0.

Memorylessness means that the future state and observations from a Markov Chain are dependent only on the current state. AKA the current state contains all the information about the system.

$$\ln \Pr(e_n \dots e_1 \mid E_0) = \ln \Pr(e_1 \mid E_0) + \sum_{i=2}^{n} \ln \Pr(e_i \mid e_{i-1})$$

Klaus Hasselmann, who won the 2021 Nobel Prize, studied climate models in which short-term random fluctuations can have longer-term effects. Suppose we're given a dataset $(y_0, y_1, \dots, y_n)$ of temperatures at timepoints $t = 0, 1, \dots n$, and we use a Hasselmann-style model,

$$Y_t = \alpha + \beta \sin(2\pi\omega t) + \gamma t + E_t$$

where $(E_0, E_1, \dots)$ is a Markov Chain as described above, and α, β and γ are unknown parameters to be estimated.

(c) Give an expression for the log likelihood of $(y_1, \dots, y_n)$, given the value for Y_0.

$$\begin{aligned}\Pr(Y_{t+1} \mid Y_t) &= \Pr(E_{t+1} = Y_{t+1} - \alpha - \beta \sin(2\pi\omega(t+1)) - \gamma(t+1) \mid E_t = Y_t - \alpha - \beta \sin(2\pi\omega(t)) - \gamma t) \\ &= \frac{1}{\sqrt{2\pi}(1-\lambda)\sigma} e^{-\frac{(Y_{t+1} - Y_t - \beta \sin(2\pi\omega(t+1)) - \beta \sin(2\pi\omega t) - \gamma)^2}{2(1-\lambda)^2\sigma^2}}\end{aligned}$$

Using memorylessness:

$$\Pr(y_1, \dots, y_n \mid Y_0) = \Pr(y_1 \mid Y_0) \cdot \prod_{t=1}^{n-1} \Pr(y_{t+1} \mid y_t)$$

Therefore the log likelihood is given by:

$$\begin{aligned}\ln \Pr(y_1, \ldots, y_n \mid Y_0) &= \ln \Pr(y_1 \mid Y_0) + \sum_{t=1}^{n-1} \ln \Pr(y_{t+1} \mid y_t) \\ &= -\frac{n}{2}\ln(2\pi) - n\ln\sigma - \frac{(y_0 - Y_0 - \beta\sin 2\pi\omega - \gamma)^2}{2(1-\lambda)^2\sigma^2} \\ &\quad - \sum_{t=1}^{n-1} \frac{(y_{t+1} - y_t - \beta\sin(2\pi\omega(t+1)) - \beta\sin(2\pi\omega t) - \gamma)^2}{2(1-\lambda)^2\sigma^2}\end{aligned}$$

(d) What is meant by a "linear model"? Write out a linear model that can be used to estimate the unknown parameters α, β and γ (treating all other parameters as known). Identify the feature vectors.

A linear model is a model which only has linear coefficients. IE the model is a linear combination of the feature vectors. The feature vectors themselves may be nonlinear functions, however the unknown parameters can be represented as a matrix equation.

$$\forall t \in \mathbb{N}.\mathbb{E}(E_t) = 0$$

We can use this to write a formula for the expectation of Y_t:

$$\mathbb{E}(Y_t) = \alpha + \beta\sin(2\pi\omega t) + \gamma t$$

We can now rerun the experiment many times and average this to approximate the mean for Y_0, Y_1 and Y_2. Using the Central Limit Theorem, our estimates will tend towards the true values.

Using the above formula $\mathbb{E}(Y_0) = \alpha$. Therefore we can get an unbiased maximum likelihood estimator for α by averaging many Y_0.

We can form equations for Y_1 and Y_2 which we can use to give estimators for β and γ:

$$\mathbb{E}(Y_1) = \alpha + \beta\sin(2\pi\omega) + \gamma \qquad \mathbb{E}(Y_2) = \alpha + \beta\sin(4\pi\omega) + 2\gamma$$

These can be rearranged into a matrix equation:

$$\begin{pmatrix}\sin(2\pi\omega) & 1 \\ \sin(4\pi\omega) & 2\end{pmatrix}\begin{pmatrix}\beta \\ \gamma\end{pmatrix} = \begin{pmatrix}\mathbb{E}(Y_1) - \alpha \\ \mathbb{E}(Y_2) - \alpha\end{pmatrix}$$

$$\begin{pmatrix}\beta \\ \gamma\end{pmatrix} = \frac{1}{2\sin(2\pi\omega) - \sin(4\pi\omega)}\begin{pmatrix}2 & -1 \\ -\sin(4\pi\omega) & \sin(2\pi\omega)\end{pmatrix}\begin{pmatrix}\mathbb{E}(Y_1) - \alpha \\ \mathbb{E}(Y_2) - \alpha\end{pmatrix}$$