LAB 11/20

STOCHASTIC PROCESS $\overbrace{X_0, X_1, X_2, \ldots,}^{RV}$

$X_n = \{1, 2, 3\}$

EACH NODE IS $\mathbb{P}(X_{n+1} = a \mid X_n = b)$

$\{1, 2, 3\}$

$X_0 = 1 \quad \leftarrow \pi_0^T P$

$X_2 = 2 \quad \pi_0^T P^2 = \pi_1^T P$

$X_3 = 2$

$X_4 = 1$

$$\pi_n^T = \pi_0^T P^n$$

π_0 = Prob. dist. at time 0

$\pi_0 = (1, 0, 0)$

TRANSPOSE

$\pi_{n+1}^T = \pi_n^{T} \mathbb{P}$

$$[p_1, p_2, p_3] \begin{bmatrix} p_{1\to1} & p_{1\to2} & p_{1\to3} \\ p_{2\to1} & p_{2\to2} & p_{2\to3} \\ p_{3\to1} & p_{3\to2} & p_{3\to3} \end{bmatrix}$$

$= [\, p_1 p_{1\to1} + p_{2\to1} + p_3 p_{3\to1}, \quad \ldots, \quad]$

prob of being in 1 after 1 move

a

$$P = \begin{bmatrix} p_{1\to1} & p_{1\to2} & p_{1\to3} \\ p_{2\to1} & p_{2\to2} & p_{2\to3} \\ p_{3\to1} & p_{3\to2} & p_{3\to3} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{bmatrix}$$

$$\pi_{t+1}^T = \pi_t^T P$$

$\pi \cdot P$

(2)

$\lim_{n\to\infty} \pi_n = \pi_\infty \rightarrow \boxed{\pi^T_\infty = \pi^T_\infty}$ save

$\pi^T_\infty (P - I) = 0$

(a)

$\underbrace{\pi^T_\infty}_{} = \underbrace{\pi^T_\infty P}_{} \Rightarrow (P^T - I)\, \pi_\infty = 0$

$$\left[\begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.7 & 0.5 & 0.4 \\ 0.1 & 0.3 & 0.4 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right] \pi_\infty = 0$$

$$\left(\begin{array}{ccc|c} -0.8 & 0.2 & 0.2 & 0 \\ 0.7 & -0.5 & 0.4 & 0 \\ 0.1 & 0.3 & -0.6 & 0 \end{array}\right) \xrightarrow{L_1+8L_3} \left(\begin{array}{ccc|c} -0.8 & 0.2 & 0.2 & 0 \\ 0.7 & -0.5 & 0.4 & 0 \\ 0 & 2.6 & -4.6 & 0 \end{array}\right)$$

$$\xrightarrow{7L_1+8L_2} \left(\begin{array}{ccc|c} -0.8 & 0.2 & 0.2 & 0 \\ 0 & -2.6 & 4.6 & 0 \\ 0 & 2.6 & -4.6 & 0 \end{array}\right) \quad \left(\begin{array}{ccc|c} -0.8 & 0.2 & 0.2 & 0 \\ 0 & -2.6 & 4.6 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

$-2.6X_2 + 4.6X_3 = 0 \Rightarrow 2.6X_2 = 4.6X_3$

$X_2 = 1.77X_3$

$-0.8X_1 + 0.2(1.77X_3) + 0.2X_3 = 0$

$0.55X_3 = 0.8X_1 \Rightarrow X_1 = 0.69X_3$

$\pi_\infty = \begin{pmatrix} 0.69 \\ 1.77 \\ 1 \end{pmatrix}$ Normalizing by $3.46 \Rightarrow \pi_\infty = \begin{pmatrix} 0.69/3.46 \\ 1.77/3.46 \\ 1/3.46 \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.51 \\ 0.28 \end{pmatrix}$

(3)

(b)

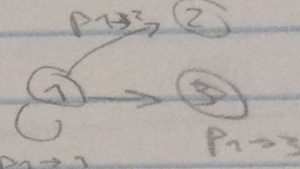
$\underbrace{T_1 \quad T_2}_{RV} \qquad T_3 = 0$

$\mu_1 = \mathbb{E}[T_1]$

$= 1 + p_{11}\mu_1 + p_{12}\mu_2 + \underbrace{p_{13}\mu_3}_{=0}$

$\downarrow$

$T_3 = 0$



$\mu_1 = 1 + p_{11}\mu_1 + p_{12}\mu_2$

$(1-p_{11})\mu_1 = 1 + p_{12}\mu_2 \Rightarrow \mu_1 = \frac{1+p_{12}\mu_2}{1-p_{11}}$

$\mu_2 = \mathbb{E}(T_2) = 1 + p_{21}\mu_1 + p_{22}\mu_2 + \underbrace{p_{23}\mu_3}_{=0}$

$(1-p_{22})\mu_2 = 1 + p_{21}\mu_1 \Rightarrow (1-p_{22})\mu_2 = 1 + p_{21}\left(\frac{1+p_{12}\mu_2}{1-p_{11}}\right)$

$(1-p_{22})\mu_2 = 1 + \frac{p_{21} + p_{12}p_{21}\mu_2}{1-p_{11}}$

$(1-p_{22})\mu_2 - \frac{p_{12}p_{21}}{1-p_{11}}\mu_2 = 1 + \frac{p_{21}}{1-p_{11}}$

$\mu_2 \underbrace{\left(1-p_{22} - \frac{p_{12}p_{21}}{1-p_{11}}\right)}_{\frac{1-p_{22}-p_{11}+p_{11}p_{22}-p_{12}p_{21}}{1-p_{11}}} = 1 + \frac{p_{21}}{1-p_{11}}$

$\mu_2 = \frac{1-p_{11}+p_{21}}{(1-p_{11})(1-p_{22})-p_{12}p_{21}} = \frac{1-0.2+0.2}{(1-0.2)(1-0.5)-0.3(0.2)} = 3.84$

$\mu_1 = \frac{1}{1-p_{11}} + \frac{p_{12}}{1-p_{11}}\left(\frac{1-p_{11}+p_{21}}{(1-p_{11})(1-p_{22})-p_{12}p_{21}}\right)$

$= \frac{1}{1-p_{11}} + \frac{p_{12}(1-p_{11}+p_{21})}{(1-p_{11})^2(1-p_{22})-(1-p_{11})(p_{12}p_{21})}$

$= \frac{1}{1-0.2} + \frac{0.3(1-0.2+0.2)}{(1-0.2)^2(1-0.5)-(1-0.2)(0.3)(0.2)}$

$= 1.25 + \frac{0.3}{0.208} = 4.67$

Lab1120

November 27, 2023

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.stats import expon
import matplotlib as mpl
import pandas as pd
```

[2]:
```
import scipy as sp
```

0.1 Exercise 1b

[18]:
```
array = np.array([1,2,3])
transition_matrix = (np.array([[0.2, 0.7, 0.1], [0.2, 0.5, 0.3], [0.2, 0.4, 0.
 ↪4]]))

x = [1]
x_iteration = 1

for i in range(30):
    if x_iteration==1:
        x_iteration=np.random.choice(array, p = transition_matrix[0])
        x.append(x_iteration)
    if x_iteration==2:
        x_iteration=np.random.choice(array, p = transition_matrix[1])
        x.append(x_iteration)
    if x_iteration==3:
        x_iteration=np.random.choice(array, p = transition_matrix[2])
        x.append(x_iteration)
```

[9]:
```
#np.transpose(pi)@transition_matrix
```

[9]:
```
matrix([[0.2, 0.7, 0.1]])
```

[19]:
```
x
```

[19]:
```
[1,
 2,
```

```
3,
2,
2,
1,
1,
2,
2,
2,
2,
1,
2,
3,
2,
3,
1,
2,
2,
1,
2,
1,
2,
2,
3,
2,
3,
3,
3,
3,
2,
1,
2,
1,
2,
2,
1,
2,
1,
2,
1,
2,
2,
2,
3,
2,
3,
2,
1]
```

0.2 Exercise 2b

[3]:
```
array = np.array([1,2,3])
transition_matrix = (np.array([[0.2, 0.7, 0.1], [0.2, 0.5, 0.3], [0.2, 0.4, 0.
↪4]]))

pi = [0,0,1]
result_2b = np.transpose(pi)@np.linalg.matrix_power(transition_matrix, 100)
result_2b
```

[3]:
```
array([0.2       , 0.51111111, 0.28888889])
```

We can see we got the same results as in the handwritten part

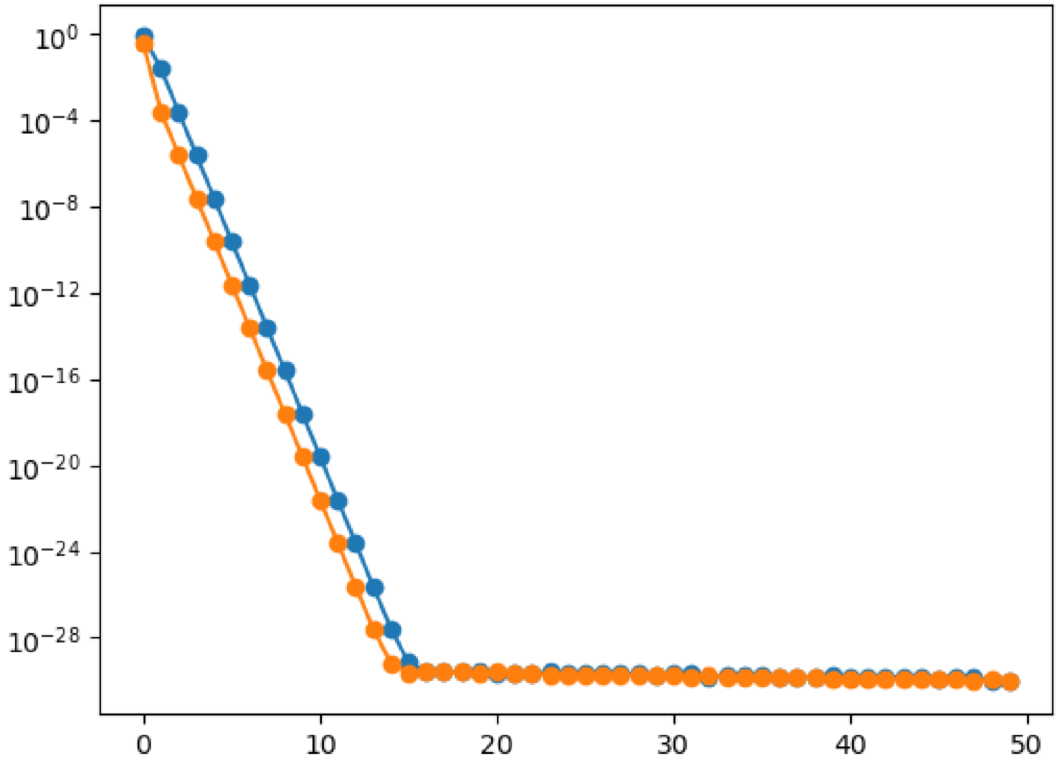
[15]:
```
x_0 = [0,0,1]
list_x0 = []
x_1 = [0,1,0]
list_x1 = []

def norm2(x):
    return(np.sum(x**2) ** 0.5)

for i in range(50):
    #Source consulted: https://stackoverflow.com/questions/35213592/
↪numpy-calculate-square-of-norm-2-of-vector
    iterate_0 = (norm2((np.transpose(x_0)@np.linalg.
↪matrix_power(transition_matrix, i))- (np.transpose(x_0)@np.linalg.
↪matrix_power(transition_matrix, 100)))**2)
    list_x0.append([i,iterate_0])
    iterate_1 = (norm2((np.transpose(x_1)@np.linalg.
↪matrix_power(transition_matrix, i))- (np.transpose(x_1)@np.linalg.
↪matrix_power(transition_matrix, 100)))**2)
    list_x1.append([i,iterate_1])

df_x0 = pd.DataFrame(list_x0)
df_x0=df_x0.rename(columns={0: "i", 1: "(|| _i -  _ω||)^2"})
df_x1 = pd.DataFrame(list_x1)
df_x1=df_x1.rename(columns={0: "i", 1: "(|| _i -  _ω||)^2"})
```

[16]:
```
plt.plot(df_x0['i'], df_x0["(|| _i -  _ω||)^2"],'o-');
plt.plot(df_x1['i'], df_x1["(|| _i -  _ω||)^2"],'o-');
plt.yscale('log')
```



We can see that after around 14 iterations both cases converge, and that before the norm square of [0,0,1] is greater than the norm square of [0,1,0]

0.3 Exercise 3 a

We create a function from the code of question 1a changed

[27]:
```
def get_markov_chain(input_array, input_transition_matrix, x_start):
    time_saved = []
    for i in range(10000):
        x_iteration = x_start
        time = 0
        while x_iteration!=3:
            if x_iteration==1:
                x_new=np.random.choice(input_array, p =␣
↪input_transition_matrix[0])
            elif x_iteration==2:
                x_new=np.random.choice(input_array, p =␣
↪input_transition_matrix[1])
            elif x_iteration==3:
                x_new=np.random.choice(input_array, p =␣
↪input_transition_matrix[2])
```

```
            time+=1
            x_iteration=x_new
        time_saved.append(time)
    return time_saved
```

[28]:
```
array = np.array([1,2,3])
transition_matrix = (np.array([[0.2, 0.7, 0.1], [0.2, 0.5, 0.3], [0.2, 0.4, 0.
 ↪4]]))
```

[29]:
```
x_1 = get_markov_chain(array, transition_matrix,1)
x_2 = get_markov_chain(array, transition_matrix,2)
```

[26]:
```
np.mean(x_1)
```

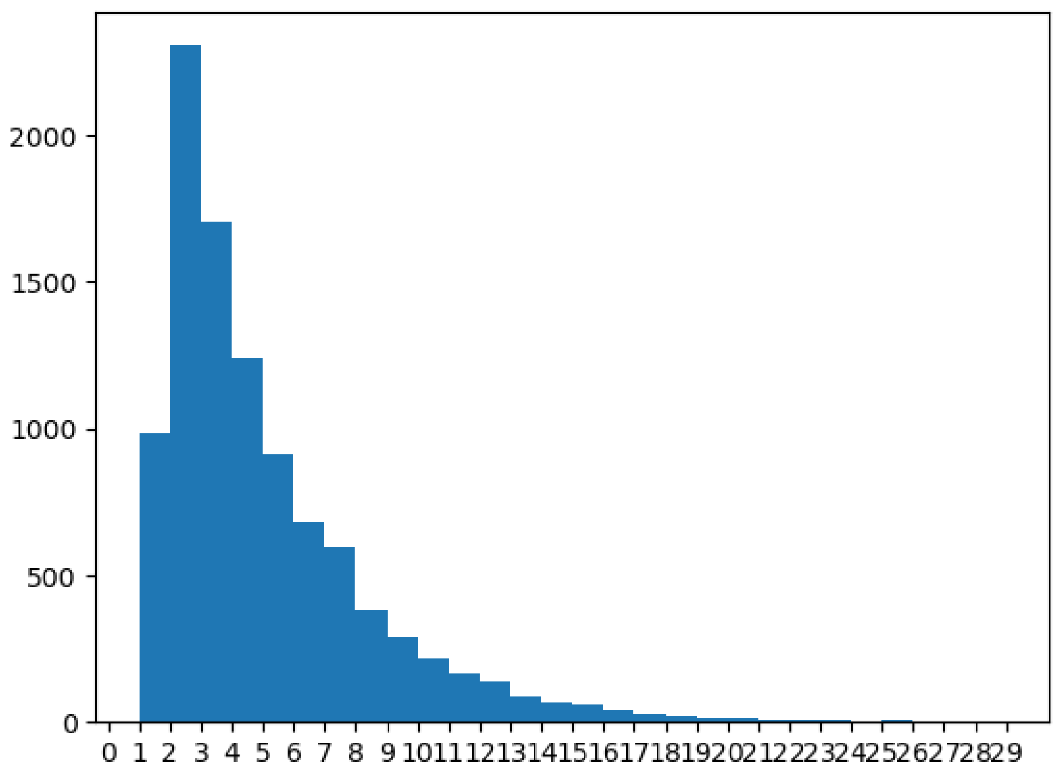
[26]: 4.6017

[25]:
```
np.mean(x_2)
```

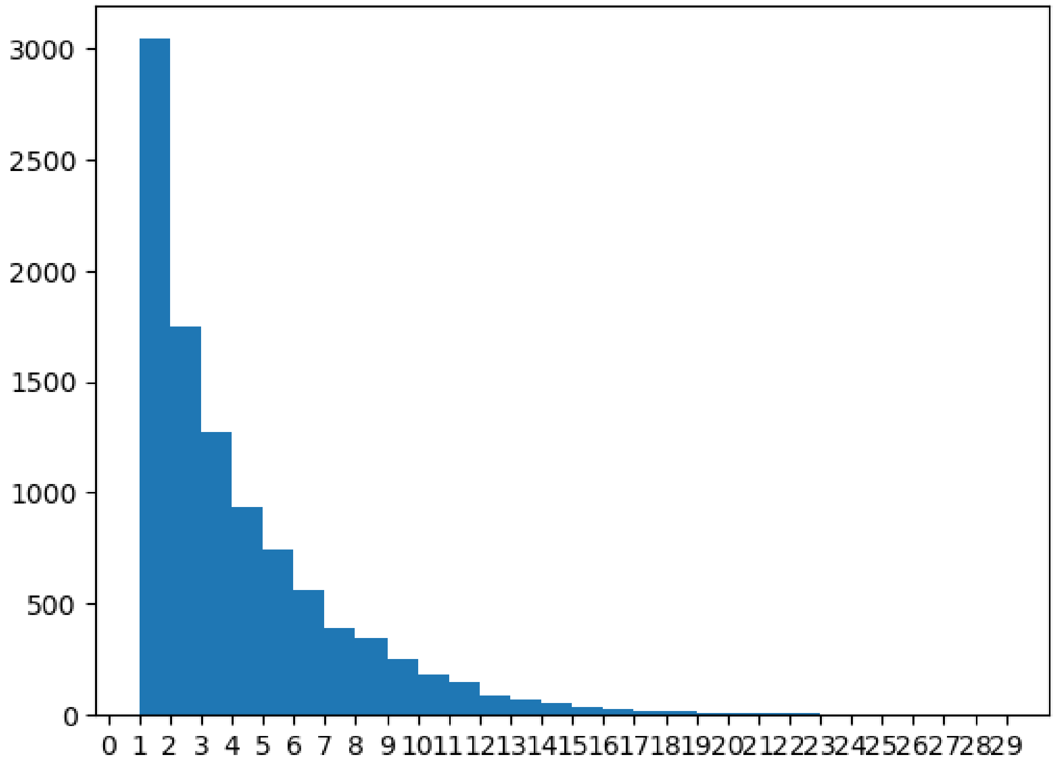
[25]: 3.8665

We can see we got the same results as in the handwritten part

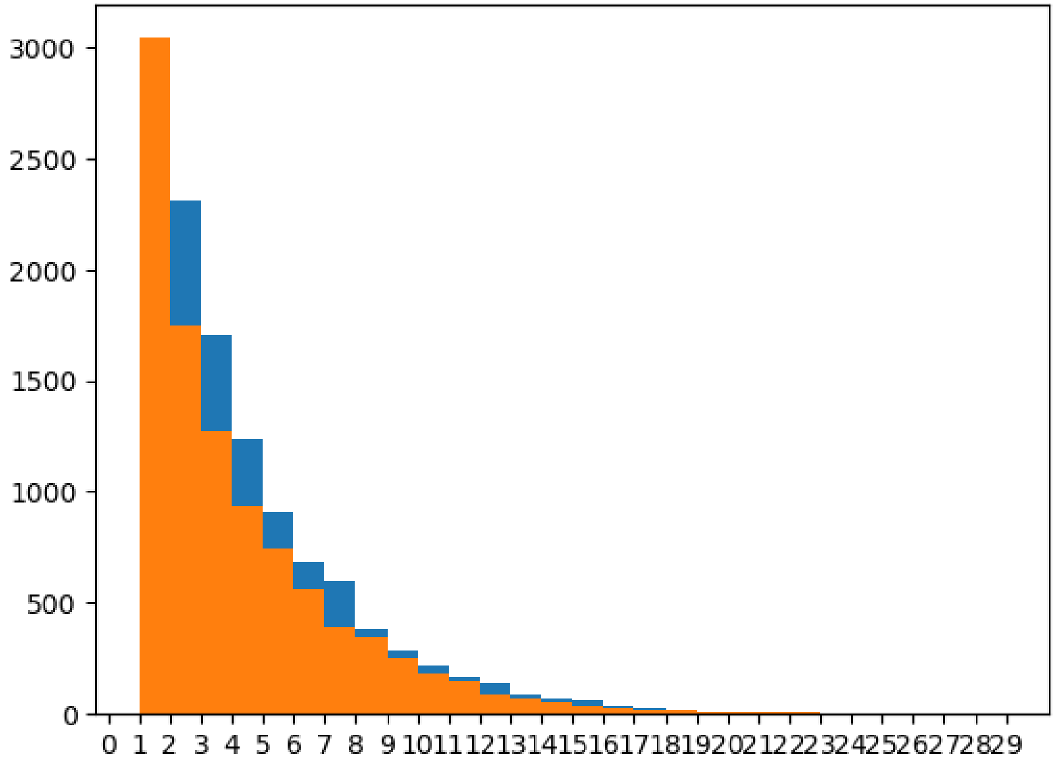
[36]:
```
plt.hist(x_1, bins=np.arange(1,30,1)); #, bins=100, density=True
plt.xticks(range(30));
```



[38]:
```
plt.hist(x_2, bins=np.arange(1,30,1)); #, bins=100, density=True
plt.xticks(range(30));
```



[40]:
```
plt.hist(x_1, bins=np.arange(1,30,1)); #, bins=100, density=True
plt.hist(x_2, bins=np.arange(1,30,1)); #, bins=100, density=True
plt.xticks(range(30));
```


3000
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29