Sina Mahbobi

March 11, 2020

1 EE511 Project 5

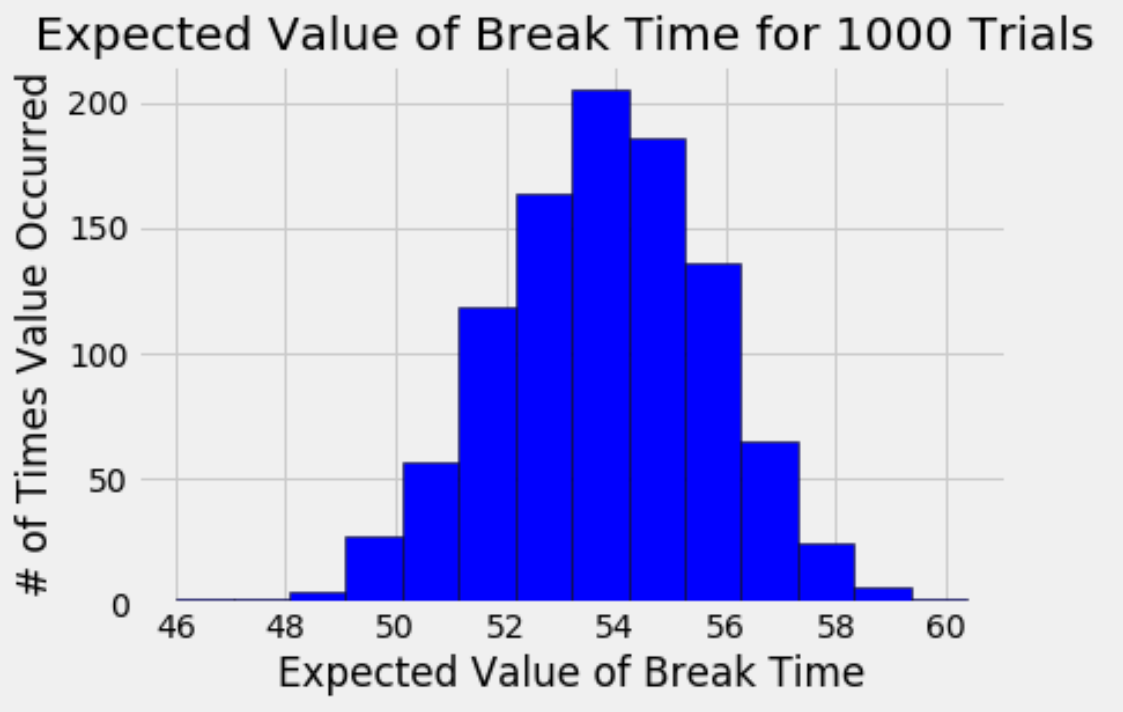
1.0.1 Question 1

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import random
import math
```

[47]:
```
break_time_arr = np.zeros(1000)
for i in range(0,1000):
    mean = 1/25 #mean service time
    index = 0
    na = 0 # number of arrivals
    nd = 0 # number of departures
    N = 0 # number of customers
    t = 0 # current time
    T = 100; # Total hours of operation
    service_time = 0 # Service time of one job
    total_break_time = 0 # Total break time
    service_comp_time = math.inf  # time after completion of current service

    time_array = np.empty(100000)
    time_array[index] = np.random.exponential(1/4)
    arr_rates = [4,7,10,13,16,19,16,13,10,7] #Variable poisson rates

    #print(time_array.shape)
    #print(time_array)

    while(time_array[index] < T):
        lamda = 19
        time = time_array[index]
        while (time < T):
            u1 = random.random()
            time = time - math.log(u1)/lamda
            u2 = random.random()
            modulus_time = (int(time) % 10) - 1
            #print(modulus_time)
```

```
            comp_time = arr_rates[modulus_time+1]/lamda
            if (u2 <= comp_time):
                time_array[index+1] = time
                index = index + 1
                break
    index = 0
    total_break = time_array[index]
    arrival_time = time_array[index]

    while(arrival_time <= T or N > 0): #Continue serving until no more
        #customers arrive or queue has been cleared
        #Case 1
        if((arrival_time <= service_comp_time) and (arrival_time <=T)):
            if(t < arrival_time): #Update current time if only it is less than␣
↪next arrival
                t=arrival_time
            na += 1
            N += 1
            if(index == len(time_array)-1):
                arrival_time = time_array[index]
            else:
                arrival_time = time_array[index + 1]
            if(N==1):
                service_comp_time = t + np.random.exponential(mean)
                t = service_comp_time
            index += 1

        # Case 2
        if((service_comp_time < arrival_time) and (service_comp_time <= T)):
            N -= 1
            nd += 1
            t = service_comp_time

            break_time = 0
            if(N == 0):
                service_comp_time = math.inf

                while(t < arrival_time):

                    inc = 0.3 * random.random()
                    break_time = break_time + inc
                    t = t + inc
                total_break_time =  total_break_time+ break_time


            else:
                service_comp_time = t + np.random.exponential(mean)
```

```
                t = service_comp_time
            if (arrival_time > T):
                service_comp_time = t
            #total_break_time_arr.append(total_break_time)
        # case 3
        if(min(arrival_time,service_comp_time) > T and N >0):
            t = service_comp_time  #Update the time
            N = N - 1 #clear remaining customer
            nd +=1  #increase the departures
            service_comp_time = t+ np.random.exponential(mean) #add the service␣
 ↪times

    # terminal case

    if(min(arrival_time,service_comp_time) > T and N == 0):
        #Queue clear
        Tp = max(t-T, service_comp_time - T) #Tp is the duration for which
        #server works to clear up queue after arrivals stop
    #total_break_time_arr.append(total_break_time)
    #print(total_break_time)
    total_break_time_arr[i] = total_break_time



plt.hist(total_break_time_arr,bins = 14, edgecolor = 'black', facecolor = 'blue'␣
 ↪)
#plt.xticks(np.arange(70,120))
plt.xlabel("Expected Value of Break Time")
plt.ylabel("# of Times Value Occurred")
plt.title("Expected Value of Break Time for 1000 Trials ")
plt.style.use('fivethirtyeight')
plt.show()
print("The minimum break time is: ",str(min(total_break_time_arr)))
print("The maximum break time is: ",str(max(total_break_time_arr)))
print("The mean of the expected break time is: ",str(total_break_time_arr.
 ↪mean()))
```




```
The minimum break time is:  46.01982101404659
The maximum break time is:  60.379024485408245
The mean of the expected break time is:  53.78303269582902
```

1.0.2 Q1 Analysis

- The histogram above shows the distribution of the different expected break values from this simulation for 1000 different trials.
- The values of break time range from around 46 to 60 for the 1000 trials, and this distribution generated has a mean of around 54.

[ ]:

Sina Mahbobi

March 11, 2020

1.0.3 Question 2 Part A

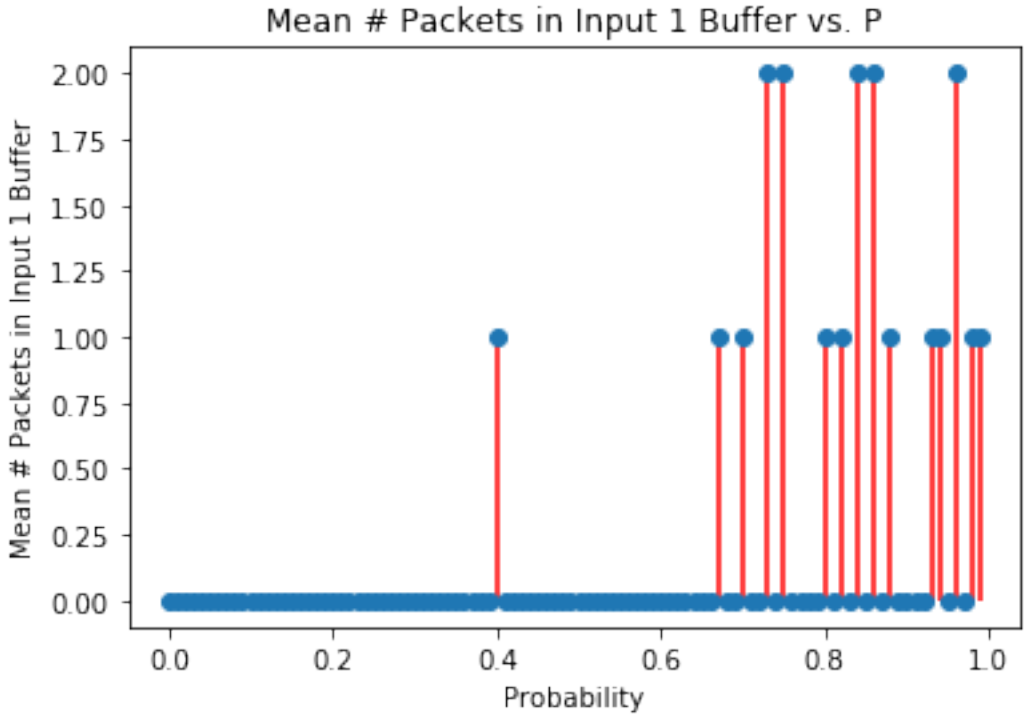
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import random
import math
```

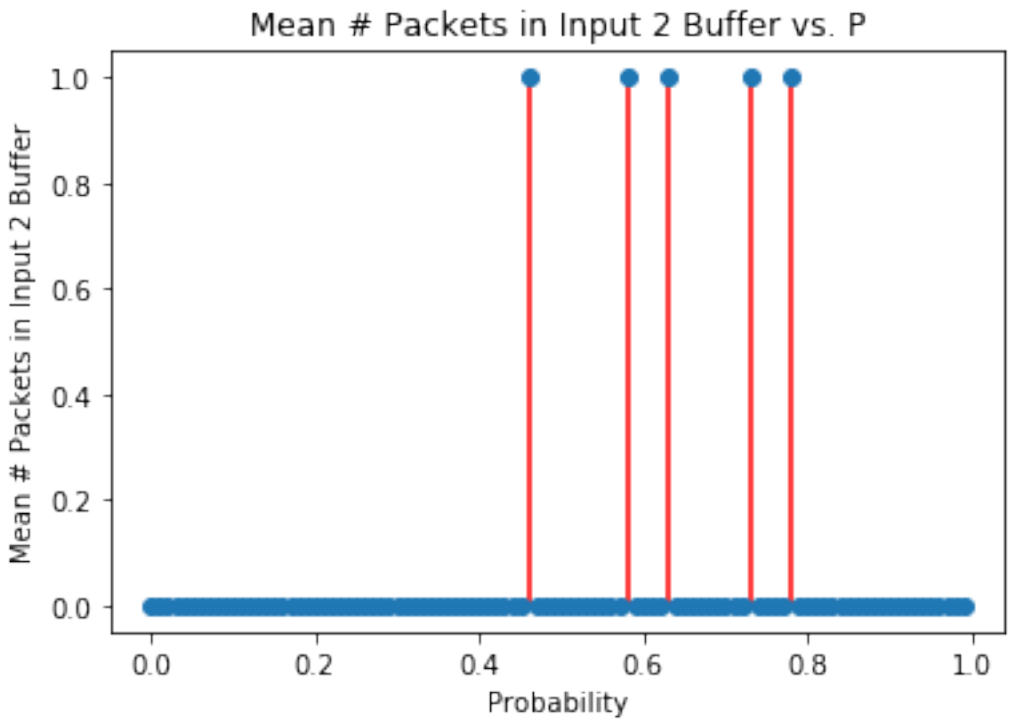
[28]:
```
r_ij = 0.5
p = np.arange(0, 1.00, 0.01)  #probability that packet arrives at port 1
num_slots = 1000 #assigning system to have 1000 slots
ip1_mean = np.empty(100)
ip2_mean = np.empty(100)
mean_packets = np.empty(100)
eff_p = np.empty(100)

for i in range(0, len(p)): #for the 100 different probabilities
    time_slot = 0
    packets = 0
    in1_buf = 0
    in2_buf = 0

    for j in range(0,1000): #for the 1000 slots in the system
        in1 = np.random.rand()
        in2 = np.random.rand()

        if(in1_buf != 0 & in2_buf == 0):

            out1 = np.random.rand()
            if(out1 < r_ij):
                out1 = 1 #output of input 1 to port 1
            else:
                out1 = 2 #output of input 1 to port 2
            if(in2 < p[i]): #checks if there is a packet at input 2
                out2 = np.random.rand()
                if(in2 < r_ij):
                    out2 = 1
                else:
                    out2 = 2
```

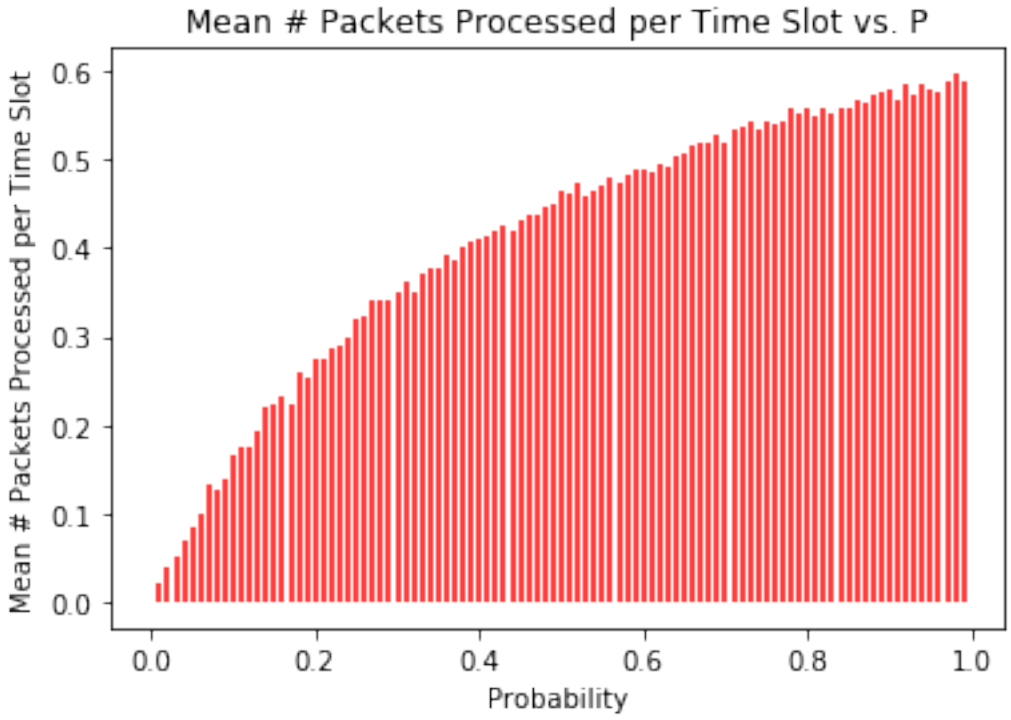
```
            else:
                out2 = 0


        elif(in1_buf == 0 & in2_buf != 0 ):
            out2 = np.random.rand()
            if(out2 < r_ij):
                out2 = 1 #output of input 2 to port 1
            else:
                out2 = 2 #output of input 2 to port 2
            if(in1 < p[i]):
                out1 = np.random.rand()
                if(out1 < r_ij):
                    out1 = 1 #output of input 1 to port 1
                else:
                    out1 = 2 #output of input 1 to port 2
            else:
                out1 = 0
        elif(in1_buf != 0 & in2_buf != 0): #both buffers have something to send
            out1 = np.random.rand()
            out2 = np.random.rand()

            if(out1 < r_ij):
                out1 = 1
            else:
                out1 = 2
            if(out2< r_ij):
                out2  = 1
            else:
                out2 = 2

            time_slot += 1
            packets += 2

            if(in1_buf !=0):
                in1_buf -= 1
            if(in2_buf != 0):
                in2_buf -= 1
        elif(in1_buf ==0 & in2_buf == 0): #both buffers empty

            if (in1 < p[i]):
                out1 = np.random.rand()
                if(out1 < r_ij):
                    out1 = 1
                else:
                    out1 = 2
            else:
```

```
                out1 = 0

            if(in2 < p[i]):
                out2 = np.random.rand()
                if(out2 < r_ij):
                    out2 = 1
                else:
                    out2 = 2
            else:
                out2 = 0

            time_slot += 1
            if(in1_buf != 0):
                in1_buf -= 1
            if(in2_buf != 0):
                in2_buf -= 1

        if(out1 != out2):
            time_slot += 1
            packets += 1
            if(in1_buf != 0):
                in1_buf -= 1
            if(in2_buf != 0):
                in2_buf -= 1
        if(out1 == out2 & out1 != 0):
            time_slot += 1
            packets += 1

            dest = np.random.rand()
            if(dest < 0.5): # if output contested the buffer chosen with 50/50␣
↪probability
                in1_buf += 1
                if(in2_buf != 0):
                    in2_buf -= 1
            else:
                in2_buf  += 1
                if(in1_buf != 0):
                    in1_buf -= 1
    ip1_mean[i] = in1_buf
    ip2_mean[i] = in2_buf
    mean_packets[i] = packets/time_slot
    eff_p[i] = packets/(2*1000)

plt.figure(1)
plt.stem(p, ip1_mean, 'r', use_line_collection = True, basefmt=" ")
plt.title('Mean # Packets in Input 1 Buffer vs. P')
plt.ylabel('Mean # Packets in Input 1 Buffer')
```

```
plt.xlabel('Probability')
plt.show()

plt.figure(2)
plt.stem(p, ip2_mean, 'r', use_line_collection = True, basefmt=" ")
plt.title('Mean # Packets in Input 2 Buffer vs. P', )
plt.ylabel('Mean # Packets in Input 2 Buffer')
plt.xlabel('Probability')
plt.show()

plt.figure(3)
plt.stem(p, mean_packets, 'r', use_line_collection = True, markerfmt=' ',␣
 ↪basefmt=" ")
plt.title('Mean # Packets Processed per Time Slot vs. P')
plt.ylabel('Mean # Packets Processed per Time Slot')
plt.xlabel('Probability')
plt.show()

print('Estimated 95% Confidence Interval for Efficiency: ('+str(eff_p[2])+', ' \
      +str(eff_p[77])+')')
```




Mean # Packets in Input 2 Buffer vs. P
Mean # Packets in Input 2 Buffer
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Probability


Mean # Packets Processed per Time Slot vs. P
Mean # Packets Processed per Time Slot
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Probability

```
Estimated 95% Confidence Interval for Efficiency: (0.021, 0.483)
```

Q2 Part A Analysis

- The first graph is a stem plot of the mean number of packets in the input buffer vs. the probability that a packet arrives at port 1.
  - In this case, as the probability increases, the number of packets as processed at port 1 increases because it is more likely that there are more packets there.
- The second graph is a stem plot of the mean number of packets in the input buffer at port 2 vs. the probability that a packet arrives at port 1.
  - In this case, we see the most packets at the probabilities between 0.4 and 0.8, because there is high uncertainty here as the whether a packet will show up at port 1 or port 2.
- The third graph is a stem plot of the mean number of packets processed per time slot vs the probability of a packet showing up at port 1. It is evident that as the probability of a packet showing up at the port increases, so does the mean number of packets processed.

[ ]:

Sina Mahbobi

March 11, 2020

1.0.4 Question 2 Part B

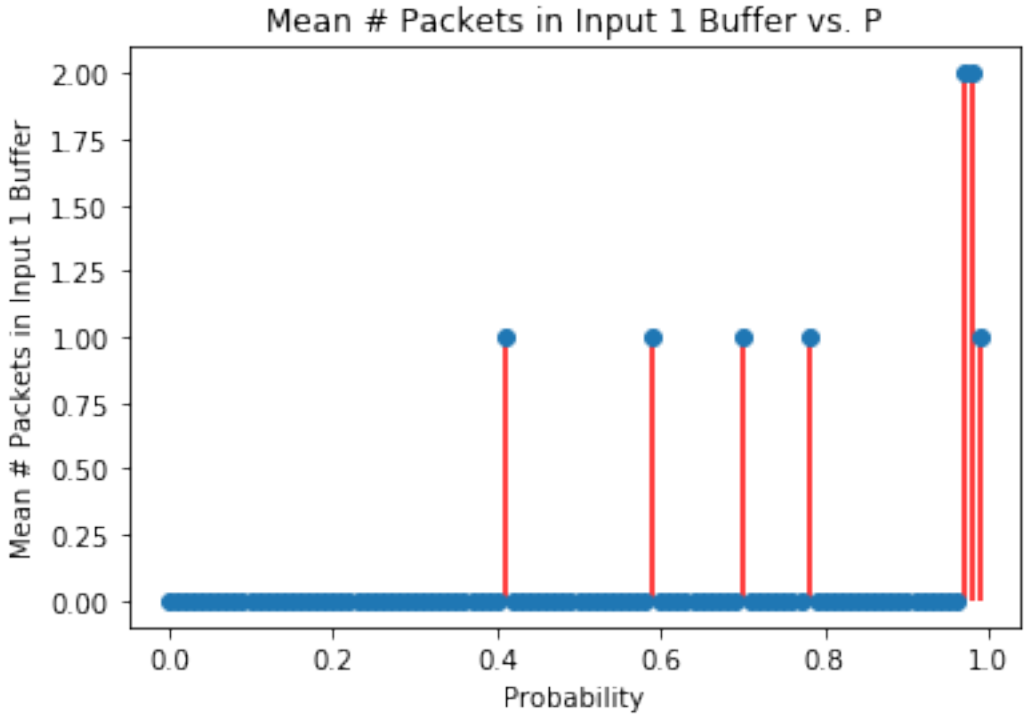
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import random
import math
```

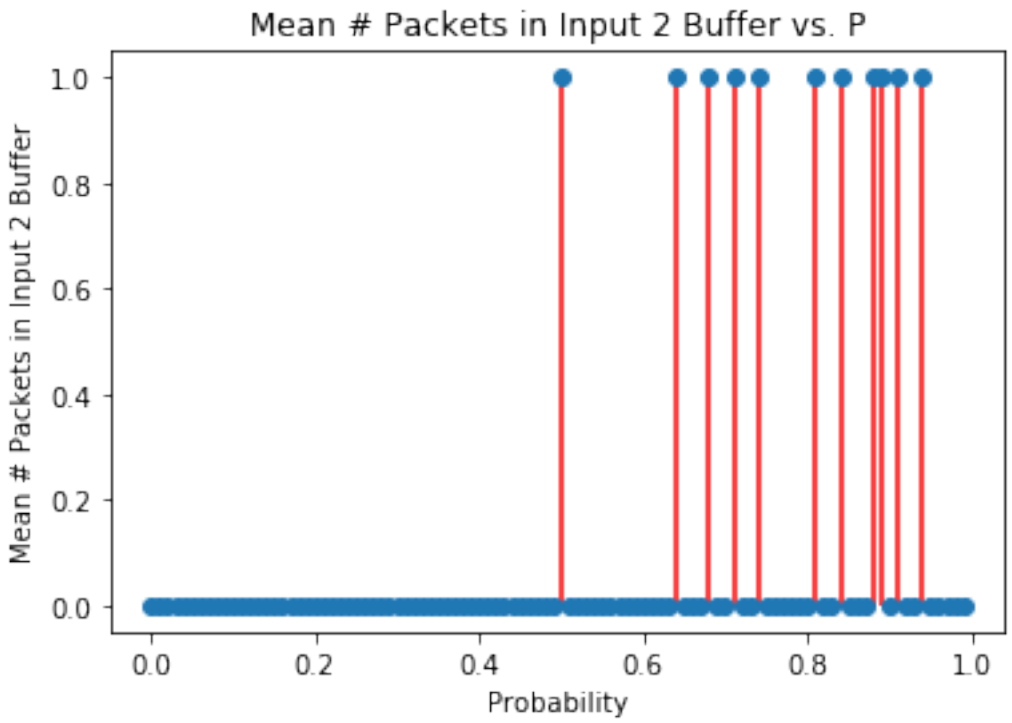
[2]:
```
r_ij = 0.75
p = np.arange(0, 1.00, 0.01)  #probability that packet arrives at port 1
num_slots = 1000 #assigning system to have 1000 slots
ip1_mean = np.empty(100)
ip2_mean = np.empty(100)
mean_packets = np.empty(100)
eff_p = np.empty(100)

for i in range(0, len(p)): #for the 100 different probabilities
    time_slot = 0
    packets = 0
    in1_buf = 0
    in2_buf = 0

    for j in range(0,1000): #for the 1000 slots in the system
        in1 = np.random.rand()
        in2 = np.random.rand()

        if(in1_buf != 0 & in2_buf == 0):

            out1 = np.random.rand()
            if(out1 < r_ij):
                out1 = 1 #output of input 1 to port 1
            else:
                out1 = 2 #output of input 1 to port 2
            if(in2 < p[i]): #checks if there is a packet at input 2
                out2 = np.random.rand()
                if(in2 < r_ij):
                    out2 = 1
                else:
                    out2 = 2
```

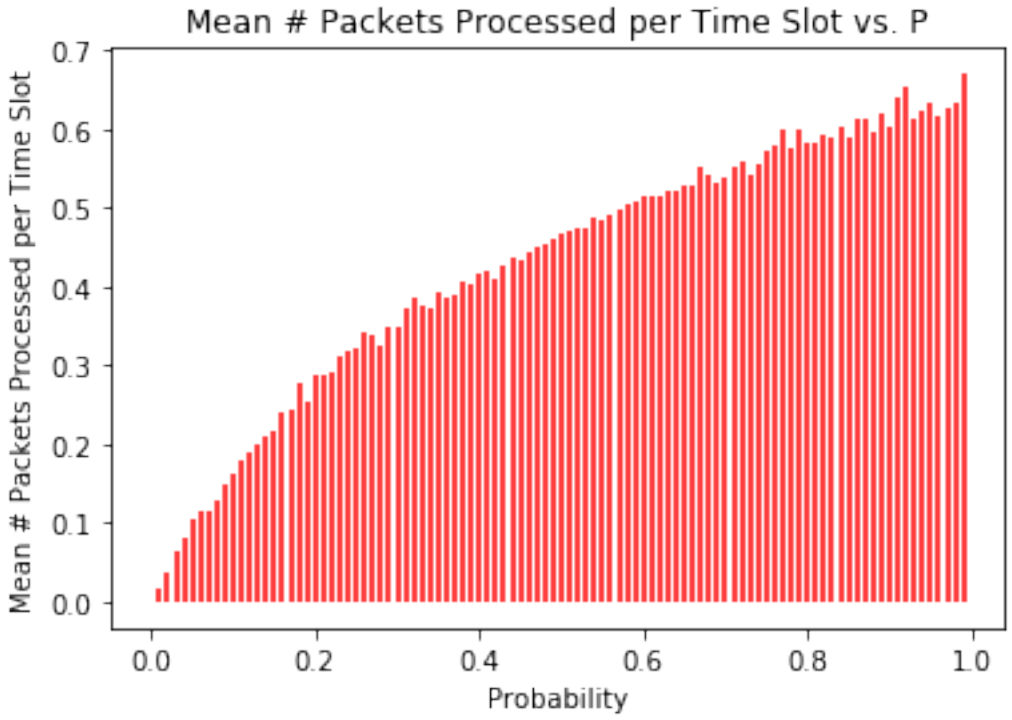
```
            else:
                out2 = 0


        elif(in1_buf == 0 & in2_buf != 0 ):
            out2 = np.random.rand()
            if(out2 < r_ij):
                out2 = 1 #output of input 2 to port 1
            else:
                out2 = 2 #output of input 2 to port 2
            if(in1 < p[i]):
                out1 = np.random.rand()
                if(out1 < r_ij):
                    out1 = 1 #output of input 1 to port 1
                else:
                    out1 = 2 #output of input 1 to port 2
            else:
                out1 = 0
        elif(in1_buf != 0 & in2_buf != 0): #both buffers have something to send
            out1 = np.random.rand()
            out2 = np.random.rand()

            if(out1 < r_ij):
                out1 = 1
            else:
                out1 = 2
            if(out2< r_ij):
                out2  = 1
            else:
                out2 = 2

            time_slot += 1
            packets += 2

            if(in1_buf !=0):
                in1_buf -= 1
            if(in2_buf != 0):
                in2_buf -= 1
        elif(in1_buf ==0 & in2_buf == 0): #both buffers empty

            if (in1 < p[i]):
                out1 = np.random.rand()
                if(out1 < r_ij):
                    out1 = 1
                else:
                    out1 = 2
            else:
```

```
                out1 = 0

            if(in2 < p[i]):
                out2 = np.random.rand()
                if(out2 < r_ij):
                    out2 = 1
                else:
                    out2 = 2
            else:
                out2 = 0

            time_slot += 1
            if(in1_buf != 0):
                in1_buf -= 1
            if(in2_buf != 0):
                in2_buf -= 1

        if(out1 != out2):
            time_slot += 1
            packets += 1
            if(in1_buf != 0):
                in1_buf -= 1
            if(in2_buf != 0):
                in2_buf -= 1
        if(out1 == out2 & out1 != 0):
            time_slot += 1
            packets += 1

            dest = np.random.rand()
            if(dest < 0.5): # if output contested the buffer chosen with 50/50␣
 ↪probability
                in1_buf += 1
                if(in2_buf != 0):
                    in2_buf -= 1
            else:
                in2_buf  += 1
                if(in1_buf != 0):
                    in1_buf -= 1
    ip1_mean[i] = in1_buf
    ip2_mean[i] = in2_buf
    mean_packets[i] = packets/time_slot
    eff_p[i] = packets/(2*1000)

plt.figure(1)
plt.stem(p, ip1_mean, 'r', use_line_collection = True, basefmt=" ")
plt.title('Mean # Packets in Input 1 Buffer vs. P')
plt.ylabel('Mean # Packets in Input 1 Buffer')
```

```
plt.xlabel('Probability')
plt.show()

plt.figure(2)
plt.stem(p, ip2_mean, 'r', use_line_collection = True, basefmt=" ")
plt.title('Mean # Packets in Input 2 Buffer vs. P', )
plt.ylabel('Mean # Packets in Input 2 Buffer')
plt.xlabel('Probability')
plt.show()

plt.figure(3)
plt.stem(p, mean_packets, 'r', use_line_collection = True, markerfmt=' ',␣
 ↪basefmt=" ")
plt.title('Mean # Packets Processed per Time Slot vs. P')
plt.ylabel('Mean # Packets Processed per Time Slot')
plt.xlabel('Probability')
plt.show()

print('Estimated 95% Confidence Interval for Efficiency: ('+str(eff_p[2])+', ' \
      +str(eff_p[77])+')')
```




Mean # Packets in Input 2 Buffer vs. P
Mean # Packets in Input 2 Buffer
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Probability


Mean # Packets Processed per Time Slot vs. P
Mean # Packets Processed per Time Slot
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Probability

```
Estimated 95% Confidence Interval for Efficiency: (0.02, 0.484)
```

Q2 Part B Analysis

- Now the probability of a packet switching from i to j is 0.75, and 0.25 for j to i so it is no longer symmetrical.
- The first graph is a stem plot of the mean number of packets in the input buffer vs. the probability that a packet arrives at port 1.
  - In this case, as the probability increases, the number of packets as processed at port 1 increases because it is more likely that there are more packets there.
- The second graph is a stem plot of the mean number of packets in the input buffer at port 2 vs. the probability that a packet arrives at port 1.
  - In this case, an increasing p results in more packets in the buffer at port 2.
- The third graph is a stem plot of the mean number of packets processed per time slot vs the probability of a packet showing up at port 1. It is evident that as the probability of a packet showing up at the port increases, so does the mean number of packets processed.

[ ]:

1.0.5 Question 3

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb
```

[17]:
```
index = 200
N = 100
#convergence = 10000
variations = [0, 1, 10, 25, 50, 100, 150, 200]
# 0: 0 A1, all A2
# 1: 1 A1, 199 A2
# 10: 10 A1, 190 A2
# 25: 25 A1, 175 A2
# 50: 50 A1, 150 A2
# 100: 100 A1, 100 A2
# 150: 150 A1, 50 A2
# 200: 200 A1, 0 A2

for trial in range (0,len(variations)):
    conv_100_trials = np.zeros(100)

    initial_input = np.zeros(201)
    n = variations[trial]
    initial_input[n] = 1
    #print(initial_input)
    P = np.zeros((2*N+1, 2*N+1))
    #intermediate = np.zeros(convergence, index + 1);
```

```python
    #stores all intermediate state probabilities till convergence
    for i in range(0,index+1):
        for j in range(0, index+1):
            P[i,j] = (comb(2*N,j, exact=True, repetition=False)*(((i)/
↪(2*N))**(j))*
                        ((1-(i)/(2*N))**(2*N-j)) )

    n = 3000 # number of steps to take
    output = np.zeros((n+1,2*N+1)) # clear out any old values
    t = np.arange(0,n)
    output[0,:] = initial_input  #generate first output value
    for i in range(1,n):
        output[i,:] = np.dot(output[i-1,:],P)
        if np.allclose(output[i,:],output[i-1,:]):



            break

    print('Initial state: A1 = ' + str(variations[trial]) + ',' \
          + ' A2 = ' + str(200 - variations[trial]))
    print('Mean Convergence for 100 trials after '+str(i),'iterations')
    print('Initial state vector x(t):')
    print(output[0,:])
    print('\n')
    print('Steady state vector x(t) after '+str(i)+' steps:  ')
    print(output[i,:])
    print('\n')
```

```
Initial state: A1 = 0, A2 = 200
Mean Convergence for 100 trials after 1 iterations
Initial state vector x(t):
[1. 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. 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. 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.]


Steady state vector x(t) after 1 steps:
[1. 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. 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. 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.]


Initial state: A1 = 1, A2 = 199
Mean Convergence for 100 trials after 1421 iterations
Initial state vector x(t):
[0. 1. 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. 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. 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.]


Steady state vector x(t) after 1421 steps:
[9.94988089e-01 9.82913577e-08 1.10268044e-07 1.13727099e-07
 1.15387889e-07 1.16437869e-07 1.17145412e-07 1.17652314e-07
 1.18035458e-07 1.18336137e-07 1.18578709e-07 1.18778749e-07
 1.18946714e-07 1.19089885e-07 1.19213477e-07 1.19321330e-07
 1.19416330e-07 1.19500694e-07 1.19576150e-07 1.19644065e-07
 1.19705540e-07 1.19761466e-07 1.19812577e-07 1.19859478e-07
 1.19902677e-07 1.19942603e-07 1.19979616e-07 1.20014029e-07
 1.20046106e-07 1.20076080e-07 1.20104149e-07 1.20130490e-07
 1.20155255e-07 1.20178580e-07 1.20200585e-07 1.20221376e-07
 1.20241047e-07 1.20259685e-07 1.20277364e-07 1.20294153e-07
 1.20310114e-07 1.20325303e-07 1.20339771e-07 1.20353562e-07
 1.20366720e-07 1.20379283e-07 1.20391285e-07 1.20402759e-07
 1.20413735e-07 1.20424239e-07 1.20434296e-07 1.20443931e-07
 1.20453162e-07 1.20462012e-07 1.20470497e-07 1.20478635e-07
 1.20486442e-07 1.20493932e-07 1.20501120e-07 1.20508016e-07
 1.20514635e-07 1.20520986e-07 1.20527080e-07 1.20532927e-07
 1.20538535e-07 1.20543914e-07 1.20549071e-07 1.20554014e-07
 1.20558749e-07 1.20563284e-07 1.20567624e-07 1.20571775e-07
 1.20575743e-07 1.20579532e-07 1.20583148e-07 1.20586595e-07
 1.20589877e-07 1.20592998e-07 1.20595962e-07 1.20598772e-07
 1.20601431e-07 1.20603943e-07 1.20606311e-07 1.20608536e-07
 1.20610621e-07 1.20612570e-07 1.20614383e-07 1.20616062e-07
 1.20617610e-07 1.20619029e-07 1.20620318e-07 1.20621481e-07
 1.20622518e-07 1.20623429e-07 1.20624217e-07 1.20624882e-07
 1.20625424e-07 1.20625845e-07 1.20626143e-07 1.20626321e-07
```

```
 1.20626378e-07 1.20626314e-07 1.20626128e-07 1.20625822e-07
 1.20625394e-07 1.20624844e-07 1.20624172e-07 1.20623377e-07
 1.20622457e-07 1.20621413e-07 1.20620243e-07 1.20618946e-07
 1.20617520e-07 1.20615964e-07 1.20614277e-07 1.20612457e-07
 1.20610501e-07 1.20608408e-07 1.20606175e-07 1.20603800e-07
 1.20601281e-07 1.20598613e-07 1.20595796e-07 1.20592825e-07
 1.20589696e-07 1.20586406e-07 1.20582952e-07 1.20579329e-07
 1.20575532e-07 1.20571556e-07 1.20567398e-07 1.20563050e-07
 1.20558508e-07 1.20553765e-07 1.20548815e-07 1.20543650e-07
 1.20538264e-07 1.20532648e-07 1.20526794e-07 1.20520692e-07
 1.20514334e-07 1.20507708e-07 1.20500803e-07 1.20493609e-07
 1.20486111e-07 1.20478297e-07 1.20470151e-07 1.20461658e-07
 1.20452801e-07 1.20443562e-07 1.20433920e-07 1.20423855e-07
 1.20413344e-07 1.20402361e-07 1.20390879e-07 1.20378869e-07
 1.20366299e-07 1.20353134e-07 1.20339335e-07 1.20324860e-07
 1.20309664e-07 1.20293695e-07 1.20276899e-07 1.20259212e-07
 1.20240567e-07 1.20220888e-07 1.20200090e-07 1.20178078e-07
 1.20154745e-07 1.20129973e-07 1.20103625e-07 1.20075548e-07
 1.20045567e-07 1.20013482e-07 1.19979063e-07 1.19942042e-07
 1.19902109e-07 1.19858903e-07 1.19811994e-07 1.19760877e-07
 1.19704944e-07 1.19643462e-07 1.19575539e-07 1.19500077e-07
 1.19415706e-07 1.19320698e-07 1.19212839e-07 1.19089240e-07
 1.18946063e-07 1.18778091e-07 1.18578046e-07 1.18335468e-07
 1.18034784e-07 1.17651634e-07 1.17144729e-07 1.16437183e-07
 1.15387202e-07 1.13726416e-07 1.10267375e-07 9.82907577e-08
 4.98808917e-03]


Initial state: A1 = 10, A2 = 190
Mean Convergence for 100 trials after 1445 iterations
Initial state vector x(t):
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 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. 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. 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.]


Steady state vector x(t) after 1445 steps:
[9.49899168e-01 8.32088304e-07 9.33477288e-07 9.62760031e-07
 9.76819500e-07 9.85708146e-07 9.91697881e-07 9.95989080e-07
 9.99232610e-07 1.00177803e-06 1.00383153e-06 1.00552498e-06
 1.00694691e-06 1.00815893e-06 1.00920521e-06 1.01011825e-06
 1.01092249e-06 1.01163669e-06 1.01227546e-06 1.01285042e-06
```

1.01337084e-06 1.01384430e-06 1.01427698e-06 1.01467403e-06
1.01503975e-06 1.01537774e-06 1.01569110e-06 1.01598242e-06
1.01625399e-06 1.01650774e-06 1.01674537e-06 1.01696836e-06
1.01717802e-06 1.01737549e-06 1.01756178e-06 1.01773780e-06
1.01790434e-06 1.01806212e-06 1.01821180e-06 1.01835393e-06
1.01848906e-06 1.01861765e-06 1.01874014e-06 1.01885690e-06
1.01896830e-06 1.01907466e-06 1.01917627e-06 1.01927342e-06
1.01936634e-06 1.01945527e-06 1.01954042e-06 1.01962199e-06
1.01970015e-06 1.01977507e-06 1.01984692e-06 1.01991582e-06
1.01998192e-06 1.02004533e-06 1.02010619e-06 1.02016458e-06
1.02022062e-06 1.02027439e-06 1.02032599e-06 1.02037550e-06
1.02042298e-06 1.02046853e-06 1.02051219e-06 1.02055405e-06
1.02059414e-06 1.02063254e-06 1.02066929e-06 1.02070444e-06
1.02073804e-06 1.02077013e-06 1.02080075e-06 1.02082994e-06
1.02085773e-06 1.02088416e-06 1.02090926e-06 1.02093306e-06
1.02095558e-06 1.02097685e-06 1.02099690e-06 1.02101575e-06
1.02103342e-06 1.02104992e-06 1.02106527e-06 1.02107950e-06
1.02109262e-06 1.02110463e-06 1.02111556e-06 1.02112541e-06
1.02113420e-06 1.02114192e-06 1.02114860e-06 1.02115424e-06
1.02115884e-06 1.02116241e-06 1.02116495e-06 1.02116646e-06
1.02116695e-06 1.02116641e-06 1.02116485e-06 1.02116227e-06
1.02115866e-06 1.02115401e-06 1.02114833e-06 1.02114160e-06
1.02113383e-06 1.02112500e-06 1.02111510e-06 1.02110413e-06
1.02109207e-06 1.02107891e-06 1.02106464e-06 1.02104923e-06
1.02103269e-06 1.02101498e-06 1.02099608e-06 1.02097599e-06
1.02095467e-06 1.02093210e-06 1.02090826e-06 1.02088311e-06
1.02085663e-06 1.02082880e-06 1.02079956e-06 1.02076890e-06
1.02073676e-06 1.02070312e-06 1.02066792e-06 1.02063113e-06
1.02059268e-06 1.02055254e-06 1.02051064e-06 1.02046693e-06
1.02042134e-06 1.02037381e-06 1.02032426e-06 1.02027262e-06
1.02021880e-06 1.02016271e-06 1.02010427e-06 1.02004338e-06
1.01997991e-06 1.01991377e-06 1.01984482e-06 1.01977294e-06
1.01969797e-06 1.01961976e-06 1.01953815e-06 1.01945295e-06
1.01936398e-06 1.01927101e-06 1.01917382e-06 1.01907216e-06
1.01896575e-06 1.01885431e-06 1.01873750e-06 1.01861497e-06
1.01848634e-06 1.01835117e-06 1.01820898e-06 1.01805926e-06
1.01790144e-06 1.01773485e-06 1.01755879e-06 1.01737245e-06
1.01717494e-06 1.01696524e-06 1.01674220e-06 1.01650452e-06
1.01625073e-06 1.01597912e-06 1.01568775e-06 1.01537435e-06
1.01503631e-06 1.01467056e-06 1.01427346e-06 1.01384073e-06
1.01336723e-06 1.01284676e-06 1.01227177e-06 1.01163295e-06
1.01091872e-06 1.01011443e-06 1.00920135e-06 1.00815503e-06
1.00694297e-06 1.00552101e-06 1.00382752e-06 1.00177398e-06
9.99228531e-07 9.95984972e-07 9.91693749e-07 9.85703997e-07
9.76815349e-07 9.62755901e-07 9.33473247e-07 8.32084676e-07
4.98991684e-02]

```
Initial state: A1 = 25, A2 = 175
Mean Convergence for 100 trials after 1478 iterations
Initial state vector x(t):
[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. 1. 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. 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. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0.]


Steady state vector x(t) after 1478 steps:
[8.74803219e-01 1.62388458e-06 1.82175302e-06 1.87890057e-06
 1.90633873e-06 1.92368562e-06 1.93537507e-06 1.94374969e-06
 1.95007971e-06 1.95504730e-06 1.95905490e-06 1.96235981e-06
 1.96513483e-06 1.96750020e-06 1.96954212e-06 1.97132400e-06
 1.97289356e-06 1.97428738e-06 1.97553402e-06 1.97665610e-06
 1.97767177e-06 1.97859577e-06 1.97944021e-06 1.98021510e-06
 1.98092884e-06 1.98158848e-06 1.98220003e-06 1.98276860e-06
 1.98329859e-06 1.98379382e-06 1.98425760e-06 1.98469281e-06
 1.98510199e-06 1.98548738e-06 1.98585096e-06 1.98619449e-06
 1.98651952e-06 1.98682747e-06 1.98711958e-06 1.98739700e-06
 1.98766073e-06 1.98791170e-06 1.98815075e-06 1.98837864e-06
 1.98859606e-06 1.98880365e-06 1.98900198e-06 1.98919158e-06
 1.98937294e-06 1.98954652e-06 1.98971271e-06 1.98987191e-06
 1.99002447e-06 1.99017071e-06 1.99031093e-06 1.99044542e-06
 1.99057444e-06 1.99069822e-06 1.99081699e-06 1.99093097e-06
 1.99104035e-06 1.99114531e-06 1.99124603e-06 1.99134266e-06
 1.99143535e-06 1.99152425e-06 1.99160949e-06 1.99169118e-06
 1.99176945e-06 1.99184441e-06 1.99191614e-06 1.99198476e-06
 1.99205035e-06 1.99211299e-06 1.99217277e-06 1.99222975e-06
 1.99228401e-06 1.99233560e-06 1.99238461e-06 1.99243107e-06
 1.99247504e-06 1.99251658e-06 1.99255572e-06 1.99259252e-06
 1.99262702e-06 1.99265924e-06 1.99268923e-06 1.99271701e-06
 1.99274263e-06 1.99276609e-06 1.99278743e-06 1.99280668e-06
 1.99282384e-06 1.99283894e-06 1.99285199e-06 1.99286301e-06
 1.99287200e-06 1.99287898e-06 1.99288396e-06 1.99288693e-06
 1.99288790e-06 1.99288688e-06 1.99288385e-06 1.99287882e-06
 1.99287179e-06 1.99286274e-06 1.99285167e-06 1.99283857e-06
 1.99282341e-06 1.99280620e-06 1.99278690e-06 1.99276550e-06
 1.99274199e-06 1.99271632e-06 1.99268848e-06 1.99265844e-06
 1.99262617e-06 1.99259162e-06 1.99255477e-06 1.99251557e-06
 1.99247398e-06 1.99242995e-06 1.99238344e-06 1.99233438e-06
 1.99228273e-06 1.99222842e-06 1.99217138e-06 1.99211156e-06
 1.99204886e-06 1.99198322e-06 1.99191455e-06 1.99184276e-06
```

```
 1.99176775e-06 1.99168943e-06 1.99160768e-06 1.99152239e-06
 1.99143344e-06 1.99134069e-06 1.99124401e-06 1.99114324e-06
 1.99103822e-06 1.99092879e-06 1.99081476e-06 1.99069593e-06
 1.99057210e-06 1.99044303e-06 1.99030849e-06 1.99016821e-06
 1.99002192e-06 1.98986931e-06 1.98971006e-06 1.98954381e-06
 1.98937018e-06 1.98918877e-06 1.98899911e-06 1.98880073e-06
 1.98859309e-06 1.98837562e-06 1.98814768e-06 1.98790857e-06
 1.98765755e-06 1.98739377e-06 1.98711630e-06 1.98682413e-06
 1.98651613e-06 1.98619105e-06 1.98584747e-06 1.98548384e-06
 1.98509839e-06 1.98468916e-06 1.98425390e-06 1.98379007e-06
 1.98329479e-06 1.98276474e-06 1.98219612e-06 1.98158453e-06
 1.98092483e-06 1.98021104e-06 1.97943610e-06 1.97859161e-06
 1.97766756e-06 1.97665184e-06 1.97552971e-06 1.97428302e-06
 1.97288915e-06 1.97131955e-06 1.96953761e-06 1.96749565e-06
 1.96513023e-06 1.96235517e-06 1.95905022e-06 1.95504258e-06
 1.95007495e-06 1.94374490e-06 1.93537025e-06 1.92368078e-06
 1.90633388e-06 1.87889575e-06 1.82174831e-06 1.62388035e-06
 1.24803219e-01]


Initial state: A1 = 50, A2 = 150
Mean Convergence for 100 trials after 1586 iterations
Initial state vector x(t):
[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. 1. 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. 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.]


Steady state vector x(t) after 1586 steps:
[7.49803683e-01 1.62005831e-06 1.81746054e-06 1.87447345e-06
 1.90184697e-06 1.91915301e-06 1.93081494e-06 1.93916985e-06
 1.94548497e-06 1.95044087e-06 1.95443905e-06 1.95773619e-06
 1.96050469e-06 1.96286451e-06 1.96490164e-06 1.96667934e-06
 1.96824522e-06 1.96963578e-06 1.97087950e-06 1.97199896e-06
 1.97301225e-06 1.97393410e-06 1.97477656e-06 1.97554965e-06
 1.97626173e-06 1.97691984e-06 1.97752997e-06 1.97809722e-06
 1.97862598e-06 1.97912007e-06 1.97958277e-06 1.98001697e-06
 1.98042521e-06 1.98080972e-06 1.98117246e-06 1.98151520e-06
 1.98183949e-06 1.98214673e-06 1.98243818e-06 1.98271496e-06
 1.98297809e-06 1.98322849e-06 1.98346700e-06 1.98369437e-06
 1.98391130e-06 1.98411841e-06 1.98431630e-06 1.98450547e-06
 1.98468643e-06 1.98485962e-06 1.98502544e-06 1.98518429e-06
```

```
 1.98533650e-06 1.98548242e-06 1.98562233e-06 1.98575652e-06
 1.98588525e-06 1.98600877e-06 1.98612728e-06 1.98624101e-06
 1.98635015e-06 1.98645489e-06 1.98655539e-06 1.98665181e-06
 1.98674431e-06 1.98683302e-06 1.98691808e-06 1.98699960e-06
 1.98707770e-06 1.98715250e-06 1.98722409e-06 1.98729257e-06
 1.98735802e-06 1.98742054e-06 1.98748019e-06 1.98753706e-06
 1.98759121e-06 1.98764271e-06 1.98769161e-06 1.98773799e-06
 1.98778188e-06 1.98782334e-06 1.98786241e-06 1.98789914e-06
 1.98793358e-06 1.98796574e-06 1.98799568e-06 1.98802342e-06
 1.98804899e-06 1.98807243e-06 1.98809374e-06 1.98811296e-06
 1.98813010e-06 1.98814518e-06 1.98815823e-06 1.98816924e-06
 1.98817823e-06 1.98818522e-06 1.98819020e-06 1.98819318e-06
 1.98819417e-06 1.98819317e-06 1.98819018e-06 1.98818518e-06
 1.98817818e-06 1.98816918e-06 1.98815815e-06 1.98814510e-06
 1.98813000e-06 1.98811285e-06 1.98809362e-06 1.98807229e-06
 1.98804885e-06 1.98802327e-06 1.98799551e-06 1.98796556e-06
 1.98793338e-06 1.98789894e-06 1.98786220e-06 1.98782311e-06
 1.98778164e-06 1.98773773e-06 1.98769135e-06 1.98764243e-06
 1.98759092e-06 1.98753676e-06 1.98747988e-06 1.98742021e-06
 1.98735769e-06 1.98729222e-06 1.98722373e-06 1.98715213e-06
 1.98707732e-06 1.98699920e-06 1.98691767e-06 1.98683260e-06
 1.98674388e-06 1.98665137e-06 1.98655493e-06 1.98645442e-06
 1.98634968e-06 1.98624052e-06 1.98612678e-06 1.98600825e-06
 1.98588473e-06 1.98575599e-06 1.98562178e-06 1.98548186e-06
 1.98533593e-06 1.98518370e-06 1.98502484e-06 1.98485901e-06
 1.98468581e-06 1.98450484e-06 1.98431565e-06 1.98411776e-06
 1.98391063e-06 1.98369369e-06 1.98346630e-06 1.98322779e-06
 1.98297737e-06 1.98271423e-06 1.98243744e-06 1.98214598e-06
 1.98183873e-06 1.98151442e-06 1.98117168e-06 1.98080892e-06
 1.98042440e-06 1.98001615e-06 1.97958194e-06 1.97911922e-06
 1.97862513e-06 1.97809635e-06 1.97752909e-06 1.97691895e-06
 1.97626083e-06 1.97554874e-06 1.97477564e-06 1.97393316e-06
 1.97301131e-06 1.97199800e-06 1.97087853e-06 1.96963480e-06
 1.96824423e-06 1.96667834e-06 1.96490062e-06 1.96286348e-06
 1.96050366e-06 1.95773515e-06 1.95443800e-06 1.95043981e-06
 1.94548390e-06 1.93916877e-06 1.93081385e-06 1.91915193e-06
 1.90184589e-06 1.87447237e-06 1.81745948e-06 1.62005736e-06
 2.49803683e-01]


Initial state: A1 = 100, A2 = 100
Mean Convergence for 100 trials after 1643 iterations
Initial state vector x(t):
[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. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 1. 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. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0.]


Steady state vector x(t) after 1643 steps:
[4.99803296e-01 1.62324793e-06 1.82103881e-06 1.87816398e-06
 1.90559140e-06 1.92293152e-06 1.93461641e-06 1.94298777e-06
 1.94931533e-06 1.95428100e-06 1.95828706e-06 1.96159070e-06
 1.96436465e-06 1.96672912e-06 1.96877026e-06 1.97055148e-06
 1.97212044e-06 1.97351374e-06 1.97475992e-06 1.97588159e-06
 1.97689689e-06 1.97782055e-06 1.97866468e-06 1.97943930e-06
 1.98015279e-06 1.98081220e-06 1.98142353e-06 1.98199190e-06
 1.98252172e-06 1.98301678e-06 1.98348040e-06 1.98391546e-06
 1.98432451e-06 1.98470978e-06 1.98507324e-06 1.98541666e-06
 1.98574159e-06 1.98604945e-06 1.98634147e-06 1.98661880e-06
 1.98688246e-06 1.98713336e-06 1.98737234e-06 1.98760017e-06
 1.98781753e-06 1.98802506e-06 1.98822334e-06 1.98841290e-06
 1.98859421e-06 1.98876775e-06 1.98893390e-06 1.98909307e-06
 1.98924559e-06 1.98939180e-06 1.98953200e-06 1.98966646e-06
 1.98979545e-06 1.98991921e-06 1.99003796e-06 1.99015192e-06
 1.99026129e-06 1.99036623e-06 1.99046694e-06 1.99056356e-06
 1.99065624e-06 1.99074513e-06 1.99083036e-06 1.99091205e-06
 1.99099032e-06 1.99106527e-06 1.99113700e-06 1.99120562e-06
 1.99127121e-06 1.99133385e-06 1.99139363e-06 1.99145062e-06
 1.99150488e-06 1.99155649e-06 1.99160549e-06 1.99165196e-06
 1.99169595e-06 1.99173749e-06 1.99177665e-06 1.99181346e-06
 1.99184797e-06 1.99188021e-06 1.99191021e-06 1.99193801e-06
 1.99196364e-06 1.99198712e-06 1.99200848e-06 1.99202774e-06
 1.99204493e-06 1.99206005e-06 1.99207312e-06 1.99208416e-06
 1.99209318e-06 1.99210018e-06 1.99210518e-06 1.99210818e-06
 1.99210918e-06 1.99210818e-06 1.99210518e-06 1.99210018e-06
 1.99209318e-06 1.99208416e-06 1.99207312e-06 1.99206005e-06
 1.99204493e-06 1.99202774e-06 1.99200848e-06 1.99198712e-06
 1.99196364e-06 1.99193801e-06 1.99191021e-06 1.99188021e-06
 1.99184797e-06 1.99181346e-06 1.99177665e-06 1.99173749e-06
 1.99169595e-06 1.99165196e-06 1.99160549e-06 1.99155649e-06
 1.99150488e-06 1.99145062e-06 1.99139363e-06 1.99133385e-06
 1.99127121e-06 1.99120562e-06 1.99113700e-06 1.99106527e-06
 1.99099032e-06 1.99091205e-06 1.99083036e-06 1.99074513e-06
 1.99065624e-06 1.99056356e-06 1.99046694e-06 1.99036623e-06
 1.99026129e-06 1.99015192e-06 1.99003796e-06 1.98991921e-06
 1.98979545e-06 1.98966646e-06 1.98953200e-06 1.98939180e-06
 1.98924559e-06 1.98909307e-06 1.98893390e-06 1.98876775e-06
 1.98859421e-06 1.98841290e-06 1.98822334e-06 1.98802506e-06
 1.98781753e-06 1.98760017e-06 1.98737234e-06 1.98713336e-06
 1.98688246e-06 1.98661880e-06 1.98634147e-06 1.98604945e-06
```

```
 1.98574159e-06 1.98541666e-06 1.98507324e-06 1.98470978e-06
 1.98432451e-06 1.98391546e-06 1.98348040e-06 1.98301678e-06
 1.98252172e-06 1.98199190e-06 1.98142353e-06 1.98081220e-06
 1.98015279e-06 1.97943930e-06 1.97866468e-06 1.97782055e-06
 1.97689689e-06 1.97588159e-06 1.97475992e-06 1.97351374e-06
 1.97212044e-06 1.97055148e-06 1.96877026e-06 1.96672912e-06
 1.96436465e-06 1.96159070e-06 1.95828706e-06 1.95428100e-06
 1.94931533e-06 1.94298777e-06 1.93461641e-06 1.92293152e-06
 1.90559140e-06 1.87816398e-06 1.82103881e-06 1.62324793e-06
 4.99803296e-01]


Initial state: A1 = 150, A2 = 50
Mean Convergence for 100 trials after 1586 iterations
Initial state vector x(t):
[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. 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. 1. 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.]


Steady state vector x(t) after 1586 steps:
[2.49803683e-01 1.62005736e-06 1.81745948e-06 1.87447237e-06
 1.90184589e-06 1.91915193e-06 1.93081385e-06 1.93916877e-06
 1.94548390e-06 1.95043981e-06 1.95443800e-06 1.95773515e-06
 1.96050366e-06 1.96286348e-06 1.96490062e-06 1.96667834e-06
 1.96824423e-06 1.96963480e-06 1.97087853e-06 1.97199800e-06
 1.97301131e-06 1.97393316e-06 1.97477564e-06 1.97554874e-06
 1.97626083e-06 1.97691895e-06 1.97752909e-06 1.97809635e-06
 1.97862513e-06 1.97911922e-06 1.97958194e-06 1.98001615e-06
 1.98042440e-06 1.98080892e-06 1.98117168e-06 1.98151442e-06
 1.98183873e-06 1.98214598e-06 1.98243744e-06 1.98271423e-06
 1.98297737e-06 1.98322779e-06 1.98346630e-06 1.98369369e-06
 1.98391063e-06 1.98411776e-06 1.98431565e-06 1.98450484e-06
 1.98468581e-06 1.98485901e-06 1.98502484e-06 1.98518370e-06
 1.98533593e-06 1.98548186e-06 1.98562178e-06 1.98575599e-06
 1.98588473e-06 1.98600825e-06 1.98612678e-06 1.98624052e-06
 1.98634968e-06 1.98645442e-06 1.98655493e-06 1.98665137e-06
 1.98674388e-06 1.98683260e-06 1.98691767e-06 1.98699920e-06
 1.98707732e-06 1.98715213e-06 1.98722373e-06 1.98729222e-06
 1.98735769e-06 1.98742021e-06 1.98747988e-06 1.98753676e-06
 1.98759092e-06 1.98764243e-06 1.98769135e-06 1.98773773e-06
 1.98778164e-06 1.98782311e-06 1.98786220e-06 1.98789894e-06
```

```
 1.98793338e-06 1.98796556e-06 1.98799551e-06 1.98802327e-06
 1.98804885e-06 1.98807229e-06 1.98809362e-06 1.98811285e-06
 1.98813000e-06 1.98814510e-06 1.98815815e-06 1.98816918e-06
 1.98817818e-06 1.98818518e-06 1.98819018e-06 1.98819317e-06
 1.98819417e-06 1.98819318e-06 1.98819020e-06 1.98818522e-06
 1.98817823e-06 1.98816924e-06 1.98815823e-06 1.98814518e-06
 1.98813010e-06 1.98811296e-06 1.98809374e-06 1.98807243e-06
 1.98804899e-06 1.98802342e-06 1.98799568e-06 1.98796574e-06
 1.98793358e-06 1.98789914e-06 1.98786241e-06 1.98782334e-06
 1.98778188e-06 1.98773799e-06 1.98769161e-06 1.98764271e-06
 1.98759121e-06 1.98753706e-06 1.98748019e-06 1.98742054e-06
 1.98735802e-06 1.98729257e-06 1.98722409e-06 1.98715250e-06
 1.98707770e-06 1.98699960e-06 1.98691808e-06 1.98683302e-06
 1.98674431e-06 1.98665181e-06 1.98655539e-06 1.98645489e-06
 1.98635015e-06 1.98624101e-06 1.98612728e-06 1.98600877e-06
 1.98588525e-06 1.98575652e-06 1.98562233e-06 1.98548242e-06
 1.98533650e-06 1.98518429e-06 1.98502544e-06 1.98485962e-06
 1.98468643e-06 1.98450547e-06 1.98431630e-06 1.98411841e-06
 1.98391130e-06 1.98369437e-06 1.98346700e-06 1.98322849e-06
 1.98297809e-06 1.98271496e-06 1.98243818e-06 1.98214673e-06
 1.98183949e-06 1.98151520e-06 1.98117246e-06 1.98080972e-06
 1.98042521e-06 1.98001697e-06 1.97958277e-06 1.97912007e-06
 1.97862598e-06 1.97809722e-06 1.97752997e-06 1.97691984e-06
 1.97626173e-06 1.97554965e-06 1.97477656e-06 1.97393410e-06
 1.97301225e-06 1.97199896e-06 1.97087950e-06 1.96963578e-06
 1.96824522e-06 1.96667934e-06 1.96490164e-06 1.96286451e-06
 1.96050469e-06 1.95773619e-06 1.95443905e-06 1.95044087e-06
 1.94548497e-06 1.93916985e-06 1.93081494e-06 1.91915301e-06
 1.90184697e-06 1.87447345e-06 1.81746054e-06 1.62005831e-06
 7.49803683e-01]


Initial state: A1 = 200, A2 = 0
Mean Convergence for 100 trials after 1 iterations
Initial state vector x(t):
[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. 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. 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. 1.]


Steady state vector x(t) after 1 steps:
[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. 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. 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. 1.]
```

Question 3 Analysis

- The composition of the initial population results in the same steady state vector at the end of the simulation when all of the genes are either A1 or A2.
- However, scenarios when the genes are distributed differently, violate the Perron-Frobenius theorem because the distributions at the convergence do not match in the steady state.
- If this were to not violate the Perron-Frobenius theorem and the Markov chain ergodic theorem, the steady state would result in a similar vector to the initial state.

[ ]: