ELEC373 Lab 1 Report

Adam Bayley

20176309

2022-02-18

Algorithm Code (Python)

```python
#adam bayley 20176309 19ahb

import random #for getting random numbers (RNG basically)
import matplotlib.pyplot as plt #for plotting points / lists

#Little's law:
# L (average # of items in system) = throughput(average arrival and departure rate) * Lead time

departureRate = 0.8 #mu value = .8
checkNumber = 0 #keeping track of which prob. check of the 1,000,000 its on

lambdaValues = [0.2, 0.4, 0.5, 0.6, 0.7, 0.75, 0.79, 0.795] #lambda values given in problem
W = [] #list to hold values of W after L values have been calculated
littleLawValue = [] #list to hold little law values for each lambda value
queue = [0] #list to hold probability entries for the 1,000,000 entries

def probability(arrivals, departures, checkNumber):
    # q_(k+1) = q_(k) + a_(k) - d_(k) <-- from slides with the 2 statement checks.
    #a(k) is 1 with probability lambda, 0 with probability 1-lambda
    #if q(k) + a(k) > 0, d(k) = 1 with probability mu and 0 with probability 1-mu

    nextValue = queue[checkNumber] + arrivals - departures #based on current queue value,
    if nextValue > 0:
        queue.append(nextValue) #if next value is positive, it's valid
    else:
        queue.append(0) #probability can't be negative. force it to 0.

for currentValue in lambdaValues:
    for checkNumber in range(100000):
        arrivalRV = random.uniform(0, 1) #random variable for arrival
        departureRV = random.uniform(0, 1) #random variable for departures

        #if the arrival rate is less than the lambda value, an arrival has occured. otherwise, it didn't.

        if arrivalRV < currentValue: # 1 = occured, 0 = didn't
            arrival = 1
        else:
            arrival = 0

        #similarly, if departureRV is less than mu (departure rate), a departure has occured. otherwise, it didn't.

        if departureRV < departureRate:# 1 = occured, 0 = didn't
            departure = 1
        else:
            departure = 0
        probability(arrival, departure, checkNumber) #send it back in for the next entry check

    summed = sum(queue) / len(queue) #take the sum of probabilities for this lambda value
    littleLawValue.append(summed) #append the value to the stored little law values
    queue = [0] #reset the count

print("Little Law Values:")
print(littleLawValue)
print("                ")

#calculate the delay value and append it to the list using Little's Law Calculation
for i in range(8):
    delayedValue = littleLawValue[i] / lambdaValues[i]
    W.append(delayedValue)
print("W values:")
print(W)
```

Plotting Code (Python)

```python
#plot expected queueing delay (W) with respect to the arrival rate (lambda)
plt.plot(lambdaValues, W, 'o', color = 'blue')
plt.title("Expected Queueing Delay (W) vs Arrival Rate (λ)")
plt.xlabel('Arrival Rate (λ)')
plt.ylabel("Expected Queueing Delay (W)")
plt.show()

```

Sample Output Values

```
C:\Users\secre\PycharmProjects\373assignment1\venv\Scripts\python.exe C:/Users/secre/PycharmProjects/373assignment1/test.py
Little Law Values:
[0.06610933890661093, 0.1926780732192678, 0.32788672113278866, 0.6063939360606394, 1.4044859551404485, 2.905860941390586, 10.803011969880302, 31.8440015599844]

W values:
[0.33054669453305463, 0.4816951830481695, 0.6557734422655773, 1.0106565601010657, 2.006408507343498, 3.874481255187448, 13.674698696051015, 40.05534787419421]

Process finished with exit code 0
```

Corresponding Plot