Vinay Chittam

Fet110

Communication Network

Computer Assignment

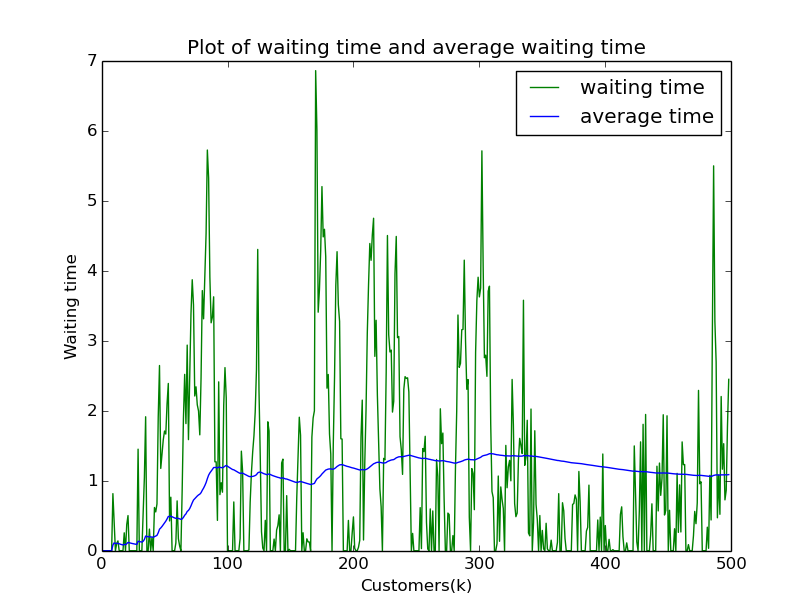
1. Let tk be the time at which kth customer arrived in the queue, WQk be the waiting time of the kth customer in the queue and Xk be the service time of kth customer. The inter-arrival time between kth customer and k+1th customer is given by Tk+1=tk+1-tk {tk+1 is the time at which k+1th customer arrived}. Given that Xk and Tk are exponential random variables with mean 1/mu and 1/lambda.

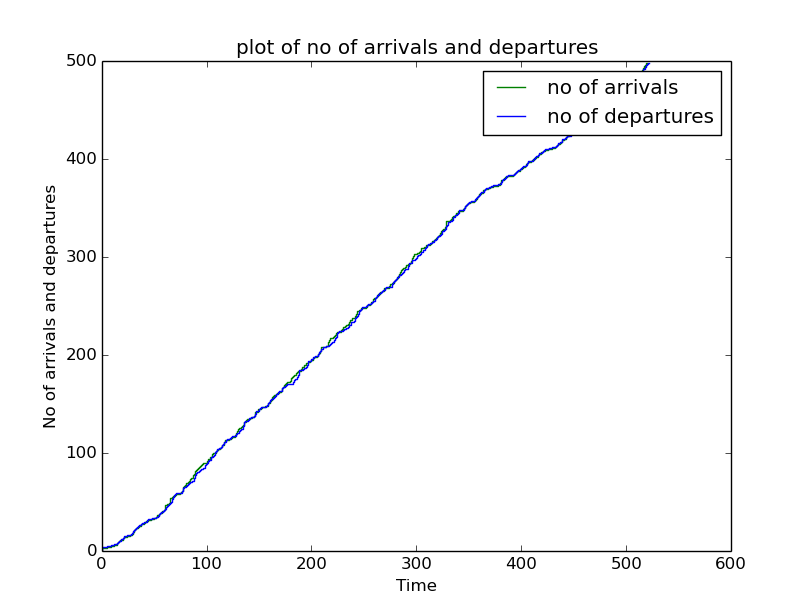
With this information we can build a recursive formula for waiting time of the customers, which is given by : WQk+1=0 if tk+1>=tk+WQk+Xk and WQk+1=tk+WQk+Xk-tk+1 if tk+1<tk+WQk+Xk

This recursion simulates the waiting times of all the customers. It can be proved as follows.

* Assume that the first customer arrives at time tk=0{here k=1 i.e. first customer}. The first customer in the system will have 0 waiting time i.e. WQk=0. This is because of the fact that he/she is the first in the system and no one else is ahead of him/her, so the customer doesn’t have to wait in the queue. The first customer will be served and the service time is Xk, which is an exponential random variable. Therefore the overall time the customer spends in the system is given by tk+WQk+Xk. The time when first customer arrived is assumed 0 i.e. tk=0, waiting time WQk =0 and the service time Xk is some exp random variable which we call Xk. So the overall time for this customer in the system is Time\_of\_firstcustomer\_insystem= tk+WQk+Xk=0+0+Xk. The time that the first customer spends in the queue= tk+WQk=0+0=0.
* The waiting time of the second customer i.e. WQk+1, can be either 0 or tk+WQk+Xk-tk+1. This depends on the inter-arrival time of the second customer, which is an exponential random variable. The arrival time of second customer can be found by adding the inter-arrival time of second customer to arrival time of first customer. If the second customer arrives after the first customer arrives, waits in the queue and gets served, then the system as well as the queue will be empty for the second customer and hence the second customer doesn’t have to wait in the queue i.e. WQk+1=0, {k=1}. If the second customer arrives before the overall time spent by the first customer in the system, then the second customer will have some waiting time in the queue. This can be clearly understood by taking an example. Let tk=5,WQk=5, Xk=5 and overall time for customer k=1 in the system is tk+WQk+Xk=15. Now assume that the second customer arrives at time tk+1=6, then the second customer has to wait in the queue till the first customer finishes his waiting time in the queue and gets served, i.e. tk+WQk+Xk {we add tk because we are calculating time in system for first customer right from his arrival point on the time scale}. But the second customer has to wait only from the point of time he has arrived in the system i.e. from the 6th minute on the time scale, thus we need to subtract this time i.e. tk+1 from tk+WQk+Xk =tk+WQk+Xk-tk+1, which accounts for the actual waiting time from the second customer, given by 5+5+5-6=9. The second customer has to wait for 9 min in the queue before the gets served. Conversely, if the second customer arrives at 16th minute i.e. after the first customer gets served, then the waiting for second customer is 0.
* The 3rd customer arrives the queue, and if the arrival time is greater than the second customer’s time spent in the system, then he will have a waiting time 0. Else he will have a waiting time which is tk+WQk+Xk-tk+1, where k=2. Therefore, this recursive formula can simulate the waiting times of all the customers in the system.

**2. I ran this program for 500 customers i.e. k=500 in the program. I redirected the output using the redirect command “>” , i.e. python runme.py > program2.doc. The waiting times, average waiting times, fraction of time when the system is empty and all the plots of problem 2 are in the program2.doc file. (didn’t post it here as it lengthy, all the waiting times and average waiting times of 500 customers.)**





#!usr/bin/env python

import numpy as np

import matplotlib.pyplot as plt

Wqi\_1=0 # The waiting time for the first customer. Since he is the first

#customer, he doesnt have to wait in the que, as there is no one ahead of

# him """

Ti\_1=0 # """ Assuming that the time at which the first customer arrived at time

# 0.Also there is no inter-arrival time for first customer """

Total=0

cus=np.arange(1,10,1)

wait=np.arange(0,5,10)

def interarrivaltimes():

lamda=1

u=np.random.uniform(0,1)

x=-np.log2(u/lamda)

return x

def interarrivaltimes1(): # this function calculates the inter-arrival times exponentially

lamda=1

tau=np.random.exponential(1)

return tau

def servicetimes(): # this function calculates the service times exponentially

X=np.random.exponential(2/3.0)

return X

fra\_time=0

k=30 # no of customers

res=[0.0]

res1=[0.0]

no\_arrivals=[0]

no\_departures=[]

timee1=[]

timee=[0]

count=0

inst=[]

kk=range(0,k-1)

for i in range(2,k):# iterating from second customer till the last customer

TAUi=interarrivaltimes1()#inter arrival time for ith customer

Ti=TAUi+Ti\_1 #arrival time of ith customer

Xi\_1=servicetimes()#service time of i-1th customer

""" Calculating waiting times """

if Ti>=Ti\_1+Wqi\_1+Xi\_1: #if the ith customer arrives after i-1th customers time spent in system, waiting time of ith customer is 0

Wqi=0

else:

Wqi=Ti\_1+Wqi\_1+Xi\_1-Ti

#if the ith customer arrives before i-1th customers time in system, he has to wait for Wqi time

res.append(Wqi)# appending each customers actial waiting time to a list

Total=Total+Wqi#finding the total waiting time for all customers

Awq=Total/i#finding the average waiting time for all the customers

res1.append(Awq)#appending average waiting times for each customer to a list

if Wqi==0:#necessary condition to find the fraction of time when the system is empty

inst\_time=Ti-(Ti\_1+Wqi\_1+Xi\_1)

times\_when\_sys\_empty=Ti-inst\_time

inst.append(times\_when\_sys\_empty)

fra\_time=fra\_time+(Ti-(Ti\_1+Wqi\_1+Xi\_1))#arrival time of ith customer - time spent by i-1th customer in the system

count=count+1

timee.append(Ti)#appending all the arrival times to a list

no\_arrivals.append(count)# appending no of arrivals to a list

no\_departures.append(count)

time\_dep\_i\_1=Ti\_1+Wqi\_1+Xi\_1

timee1.append(time\_dep\_i\_1)

Ti\_1=Ti

Wqi\_1=Wqi

customer\_no=1

print "Customer -> Waiting Time"

print "========================"

for ww in res:

print "%d\t->\t%f"%(customer\_no,ww)

customer\_no=customer\_no+1

average\_waiting\_time=sum(res1)/len(res1)#calculating average waiting time

print "the average waiting time in the queue is %f"%(average\_waiting\_time)

print Ti

print fra\_time

f=fra\_time/Ti#calculates the fraction of time when the system is empty

print "The fraction of time when the system is empty is %f"%(f)

for instances in inst:#prints all the time instances when the systems starts to be empty

print "time instances when the system is empty %f"%(instances)

plt.figure(1)

plt.plot(kk,res,'g',label='waiting time')#plots waiting time with respect to no of customers

plt.hold(True)

plt.plot(kk,res1,'b',label='average time')#plots average customers vs no of customers

plt.xlabel("Customers(k)")

plt.ylabel("Waiting time")

plt.title("Plot of waiting time and average waiting time")

plt.legend()

plt.figure(2)

plt.step(timee,no\_arrivals,'g',label="no of arrivals")

plt.hold(True)

plt.step(timee1,no\_departures,'b',label="no of departures")

plt.xlabel("Time")

plt.ylabel("No of arrivals and departures")

plt.title("plot of no of arrivals and departures")

plt.legend()

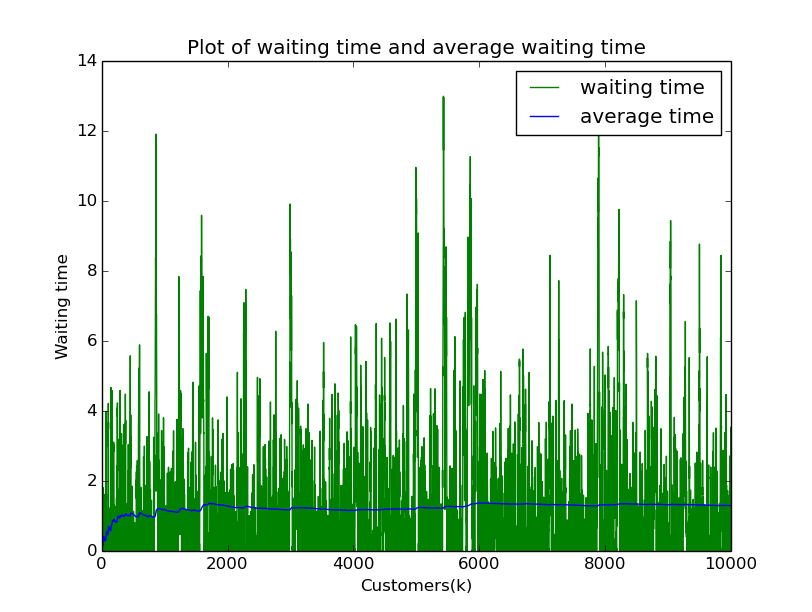
plt.show()

3. For this problem, I have run the program by taking 10,000 customers, for which the average waiting time converges to 1.227519. (mu=3/2, lambda=1)

The theoretical value is calculated by using the formula [1/(mu-lambda)]-[1/mu], where mu=3/2 and lambda =1. This results to 1.3333333.

Therefore, simulated average waiting time is close to the theoretical value. i.e. 1.227519~1.333333.

The plots are shown below:



4. This calculation is for 10,000 customers. The fraction of time when the system is empty can be theoretically calculated as follows:

The fraction of time that the server is busy is lambda/mu, therefore the fraction of time when it is free is given by 1-[lambda/mu]=1-[1/1.5]=0.33333

The simulated value of the fraction of time when the system is empty converges to 0.332925, for 10,000 customers. It is approximately close to what we expect theoretically.

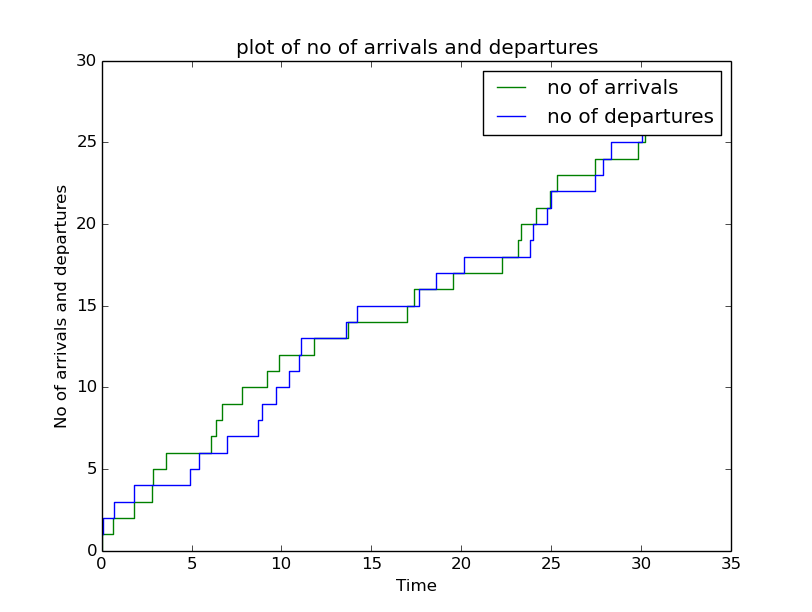
For 30 customers, the fraction of time when the system is empty is 0.305257

In order to program this calculation, I checked for the condition Wqi ==0. i.e. if the waiting time for the customer is 0, then it means the system is empty. When this condition satisfies, subtract the previous customers time of arrival (ti-1), waiting time (wqi-1) and service time (xi-1) from the arrival time of present customer (ti). i.e. ti-[(ti-1)+(wqi-1)+(xi-1)]. This gives the fraction of time when the system was empty for a particular customer. Calculate for all customer only when condition Wqi==0 meets. Then add all the fraction of times when the system was empty. Divide this entire summed fraction of time by the arrival time of the last customer. This gives us the fraction of time when the system is empty.

5. To find the number of customer arrived I found out all the inter arrival times for every customer and added them to a list. Then I incremented the counter which counts the customers as they arrive and added the counted customers to a list. Then I used the step command to plot the no of arrivals.

To find the no of departures I did the same procedure as above. But instead of adding arrival times to the list I added departure times, which can be calculated as Ti\_1+Wqi\_1+Xi\_1. After adding all the departure times calculated from above formula, I incremented the counter as the customers departed and added this to another list. Finally, I used stairs to plot them.

To find the instances of time when the system is empty, I first checked for the condition Wqi==0, i.e. if the waiting time is 0. If the condition satisfies, I calculate the exact instance of time at which the system becomes empty. In order to calculate that instance of time, I calculate the value of Ti\_1+Wqi\_1+Xi\_1 and then subtract this from Ti to get the interval during which the system will be empty. In order to find the exact instance of time when the system is empty I subtracted the time interval that I got above from the Ti. This runs in the loop and each time this value is stored in a list. Thus all the instances of time when the system is empty are captured.



The fraction of time when the system is empty is 0.390530

time instances when the system is empty 0.102547

time instances when the system is empty 0.676419

time instances when the system is empty 1.817649

time instances when the system is empty 11.105850

time instances when the system is empty 13.599472

time instances when the system is empty 14.208578

time instances when the system is empty 18.568035

time instances when the system is empty 20.118141

time instances when the system is empty 28.329977

time instances when the system is empty 30.040434

time instances when the system is empty 30.300421

time instances when the system is empty 30.732243

time instances when the system is empty 31.929593