# COMP 312 Assignment 9

Daniel Braithwaite

May 16, 2016

# 1 Python

## 1.1 Code

```python
"""(q3.py) M/M/c queueing system with monitor
   and multiple replications"""

from SimPy.Simulation import *
import random
import numpy
import math

## Useful extras —————
def conf(L):
    """confidence interval"""
    lower = numpy.mean(L) - 1.96*numpy.std(L)/math.sqrt(len(L))
    upper = numpy.mean(L) + 1.96*numpy.std(L)/math.sqrt(len(L))
    return (lower,upper)

def tablelookup(P):
    u = random.random()
    sumP = 0.0
    for i in range(len(P)):
        sumP += P[i]
        if u < sumP:
            return i

## Model —————
class Source(Process):
    """generate random arrivals"""
    def run(self, N, lamb, mu):
        for i in range(N):
            a = Arrival(str(i))
            activate(a, a.run(mu))
            t = random.expovariate(lamb)
            yield hold, self, t

class Arrival(Process):
    n = 0

    """an arrival"""
    def run(self, mu):
        arrivetime = now()

        Arrival.n += 1
        G.nummon.observe(Arrival.n)
        currentStation = 0
        while (currentStation != 3):

            station = G.stations[currentStation]#getattr(G, 'station' + str(curr
            #print station[1]
            yield request, self, station[0]
            t = random.expovariate(mu)
            yield hold, self, t
            yield release, self, station[0]

            currentStation = tablelookup(station[1])

        Arrival.n -= 1
        G.nummon.observe(Arrival.n)

        delay = now()-arrivetime
        G.delaymon.observe(delay)



class G:
    stations = [['dummy', [0, 0.1, 0.9, 0]],
                ['dummy', [0.2, 0, 0.5, 0.3]],
                ['dummy', [0, 0.1, 0, 0.9]]]
    delaymon = 'Monitor'
    nummon = 'Monitor'

def model(c, N, lamb, mu, maxtime, rvseed):
    # setup
    initialize()
    random.seed(rvseed)
    Arrival.n = 0
    G.stations[1][0] = Resource(c)
    G.stations[0][0] = Resource(c)
    G.stations[2][0] = Resource(c)

    G.delaymon = Monitor()
    G.nummon = Monitor()

    # simulate
    s = Source('Source')
    activate(s, s.run(N, lamb, mu))
    simulate(until=maxtime)

    # gather performance measures
    W = G.delaymon.mean()
    L = G.nummon.timeAverage()
    return(W, L)

## Experiment ----------

lambs = [1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2]

for lam in lambs:
    allW = []
    allL = []
    for k in range(50):
        seed = 123*k
        result = model(c=2, N=10000, lamb=lam, mu=1.0,
                       maxtime=2000000, rvseed=seed)
        allW.append(result[0])
        allL.append(result[1])

    print "\nLambda:␣", lam
    print "\tEstimate␣of␣W:",numpy.mean(allW)
    print "\tConf␣int␣of␣W:",conf(allW)
    print "\tEstimate␣of␣L:",numpy.mean(allL)
    print "\tConf␣int␣of␣L:",conf(allL)
```

## 1.2 Output

```
Lambda: 1.0
Estimate of W: 3.07115102268
Conf int of W: (3.0545014142296387, 3.087800631127156)
Estimate of L: 3.07293392046
Conf int of L: (3.0505709116845887, 3.0952969292354391)

Lambda: 1.1
Estimate of W: 3.30291442125
Conf int of W: (3.2837205998350485, 3.322108242669946)
Estimate of L: 3.63566350108
Conf int of L: (3.6075450879612228, 3.6637819142041272)

Lambda: 1.2
Estimate of W: 3.62809887072
Conf int of W: (3.6021168652532589, 3.6540808761833237)
Estimate of L: 4.36000156289
Conf int of L: (4.3220178721139915, 4.3979852536574624)

Lambda: 1.3
Estimate of W: 4.07158480319
```

```
Conf int of W: (4.0340691696071893, 4.1091004367628461)
Estimate of L: 5.30478973571
Conf int of L: (5.2468732130936129, 5.3627062583358907)

Lambda: 1.4
Estimate of W: 4.69411430471
Conf int of W: (4.6332972140664737, 4.7549313953467323)
Estimate of L: 6.57650750236
Conf int of L: (6.4790046360596518, 6.6740103686512215)

Lambda: 1.5
Estimate of W: 5.65164280855
Conf int of W: (5.5619584714599943, 5.7413271456394588)
Estimate of L: 8.49841455215
Conf int of L: (8.3475595502920541, 8.6492695539997584)

Lambda: 1.6
Estimate of W: 6.87174636875
Conf int of W: (6.7320208108733182, 7.0114719266196799)
Estimate of L: 10.9660720227
Conf int of L: (10.724309048752211, 11.207834996743063)

Lambda: 1.7
Estimate of W: 9.80069631029
Conf int of W: (9.4732894506847209, 10.12810316989798)
Estimate of L: 16.6594119506
Conf int of L: (16.06997802383087, 17.248845877432245)

Lambda: 1.8
Estimate of W: 16.76146322
Conf int of W: (15.835988850257873, 17.686937589686902)
Estimate of L: 30.1149881389
Conf int of L: (28.392912294383457, 31.837063983325834)

Lambda: 1.9
Estimate of W: 51.2452902908
Conf int of W: (45.592299283965971, 56.898281297587403)
Estimate of L: 96.2641781498
Conf int of L: (85.643801313328282, 106.88455498621624)

Lambda: 2.0
Estimate of W: 151.156888049
Conf int of W: (140.73420055369161, 161.57957554384637)
Estimate of L: 286.424293786
Conf int of L: (266.92387870816964, 305.92470886372922)
```

```
Lambda: 2.1
Estimate of W: 265.889338596
Conf int of W: (255.27250813040555, 276.50616906077812)
Estimate of L: 503.940054731
Conf int of L: (484.4649347904483, 523.4151746720172)

Lambda: 2.2
Estimate of W: 373.765552379
Conf int of W: (363.61554795108702, 383.91555680696695)
Estimate of L: 710.667473673
Conf int of L: (692.00715517220692, 729.32779217390498)
```