COMP 312 Assignment 6

Daniel Braithwaite

April 18, 2016

1 Python

1.1 Program A

1.1.1 Code

```
import random
import numpy
import math

## Useful extras ----------
def conf(x):
    """95% confidence interval"""
    lower = numpy.mean(x) - 1.96*numpy.std(x)/math.sqrt(len(x))
    upper = numpy.mean(x) + 1.96*numpy.std(x)/math.sqrt(len(x))
    return (lower,upper)

def coxian():
        """Generates a coxian random variate"""
        t = random.expovariate(2)
        if random.random() <= 0.2:
                return t

        t += random.expovariate(3)
        return t


if __name__ == "__main__":
        random.seed(123)
        a = []
        b = []
        for i in range(10000):
                variate = coxian()

                a.append(variate)
                b.append(math.pow(variate, 2))

        print "E[t]:", conf(a)
        print "E[t^2]:", conf(b)
```

1.1.2 Output

$E[t]$: (0.75741101633238683, 0.78099765992129067)
$E[t^2]$: (0.92233300526267215, 0.98510362267500762)
The theoretical values are within the 95% confidence interval which means we can conclude that $E[t]$ and $E[t^2]$ are equal to the theoretical values

1.2 Program B

1.2.1 Code

```
"""(q6.py) M/G/c queueing system with service time monitors"""

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

## Useful extras ----------
def conf(x):
    """95% confidence interval"""
    lower = numpy.mean(x) - 1.96*numpy.std(x)/math.sqrt(len(x))
    upper = numpy.mean(x) + 1.96*numpy.std(x)/math.sqrt(len(x))
    return (lower,upper)

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

class Arrival(Process):
    """an arrival"""

    n = 0

    def run(self):
        Arrival.n += 1
        G.numbermon.observe(Arrival.n)

        arrivetime = now()
        yield request, self, G.server
        t = coxian.coxian()
        G.servicemon.observe(t)
        G.servicesquaredmon.observe(t**2)
        yield hold, self, t
        yield release, self, G.server
        delay = now()-arrivetime
```

```
        G.delaymon.observe(delay)
        #print now(), "Observed delay", delay

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


class G:
    server = 'dummy'
    delaymon = 'Monitor'
    numbermon = 'Monitor'
    servicemon = 'Monitor'
    servicesquaredmon = 'Monitor'

def model(c, N, lamb, maxtime, rvseed):
    # setup
    initialize()
    random.seed(rvseed)
    G.server = Resource(c, monitored=True)
    G.delaymon = Monitor()
    G.numbermon = Monitor()
    G.servicemon = Monitor()
    G.servicesquaredmon = Monitor()

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

    # gather performance measures
    L = G.numbermon.timeAverage()
    LQ = G.server.waitMon.timeAverage()
    W = G.delaymon.mean()
    S = G.servicemon.mean()
    S2 = G.servicesquaredmon.mean()
    lambEff = L/W
    WQ = LQ/lambEff
    row = lambEff * S

    return(WQ, lambEff, S2, row)

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

allY = []

for i in range(50):
```

```
    print i
    seed = i * 123

    result = model(c=1, N=100000, lamb=1,
                   maxtime=2000000, rvseed=seed)

    WQ = result[0]
    lambEff = result[1]
    ET2 = result[2]
    row = result[3]

    allY.append(WQ - (lambEff * ET2 / (2 * (1-row))))


print "Mean Y:",numpy.mean(allY)
print "Conf Y:",conf(allY)
```

1.2.2 Output

Mean Y: -0.00233069473309
Conf Y: (-0.011739640704165668, 0.0070782512379870006)

We want Y to be 0 as this would show $w_q = \frac{\lambda E[t^2]}{2(1-\rho)}$. We see that 0 is within the confidence interval of the average Y. Not only this but our mean Y is small enough for us to conclude that $w_q = \frac{\lambda E[t^2]}{2(1-\rho)}$