Example sheet 4

Random processes

Network Performance—DJW—2010/11

Question 1. Consider the matrix of transition rates

$$R_{ij} = \begin{cases} \nu & \text{if } j = i+1 \\ i\mu & \text{if } j = i-1 \\ 0 & \text{otherwise} \end{cases}$$

for i, j in $\{0, 1, 2, \ldots, C\}$.

(i) Draw a state space diagram for the Markov process with these transition rates.

(ii) Find the equilibrium distribution ρ of this Markov process, for $\mu = 1$, $\nu = 12$ and $C = 20$.

(iii) The jump chain that corresponds to this Markov process is a Markov chain with transition probabilities

$$P_{ij} = \frac{R_{ij}}{\sum_{k=0}^{C} R_{ik}}.$$

Draw the state space diagram for the jump chain.

(iv) Find the invariant distribution π for the jump chain, with the same values of μ, ν and C as above.

(v) Comment on the difference between ρ and π.

Question 2. On the next page there are six different generators for sequences of random variables, intended to be used as flow interarrival times for a simulator of a bandwidth-sharing link. The first, `rexp(λ)`, generates a sequence of independent $\text{Exp}(\lambda)$ random variables; the others were submitted by students. Suppose the interarrival times are $X_1, X_2, \ldots$. Then we can calculate the mean arrival rate by finding

$$\frac{n}{\mathbb{E}(X_1 + X_2 + \cdots + X_n)}$$

and taking the limit as $n \to \infty$. For each of the generators listed below, find a formula for the mean arrival rate. You should validate your formula by using a computer to generate a reasonably long sequence $X_1, \ldots, X_n$ and computing $n/(X_1 + \cdots + X_n)$; repeat the computation for large enough values of n to make you confident you have computed an accurate answer. This validation is for your benefit, and you do not have to include it in your submitted answer.

Example. For generator `bursty1(λ,w)`, the code generates the sequence $X_1 = Y_1$, $X_2 = Y_2 + \cdots + Y_w$, $X_3 = Y_{w+1}$, $X_4 = Y_{w+2} + \cdots + Y_{2w}$ and so on, where each Y_i is $\text{Exp}(\lambda)$. Therefore $\mathbb{E}X_1 = 1/\lambda$, $\mathbb{E}X_2 = (w-1)/\lambda$, and so on. Thus

$$\mathbb{E}(X_1 + \cdots + X_n) = \begin{cases} \frac{n}{2}(w/\lambda) & \text{if } n \text{ even} \\ \frac{n-1}{2}(w/\lambda) + 1/\lambda & \text{if } n \text{ odd.} \end{cases}$$

When n is large, $\mathbb{E}(X_1 + \cdots + X_n)/n = w/(2\lambda)$. Hence the mean arrival rate is $2\lambda/w$. I found close agreement when I validated this formula by running

```
for n in [1000,10000,100000]:
    g = bursty1(1,5)
    x = [g.next() for i in range(n)]
    print 'n={n}, avg.rate={r}, theory={t}'.format(n=n,r=len(x)/sum(x),t=2.0/5)
```

```python
import math, random

def rexp(λ):
    while True: yield -1.0/λ * math.log(random.random())

def bursty1(λ, waittime):
    count = 0
    x = 0
    while True:
        x = x + (-1.0/λ * math.log(random.random()))
        if (count % waittime) in [0, waittime-1]:
            yield x
            x = 0
        count = count+1

def bursty2(λ):
    burst, add, curr = 0,0,0
    while True:
        burst += 1
        if burst==3:
            add, burst = curr*λ,0
        else:
            add = 0
        curr = (-1.0/λ * math.log(random.random()))
        yield curr+add

def bursty3(λ, p=2):
    r1 = λ*(p+1)/2.0
    r2 = r1/p
    while True:
        yield -1.0/r1 * math.log(random.random())
        yield -1.0/r2 * math.log(random.random())

def bursty4(λ):
    a = False
    while True:
        a = not a
        r = λ*3/2.0 if a else λ*3/4.0
        yield -1.0/r * math.log(random.random())

def bursty5(λ, fr, bi, bl=2):
    while True:
        if random.random()<=fr:
            for i in range(bl): yield bi
        else:
            for i in range(bl): yield -1.0/λ * math.log(random.random())
```