STAT 280   October 31, 2001

Sec 7.5 The M/M/1 Queuing Process

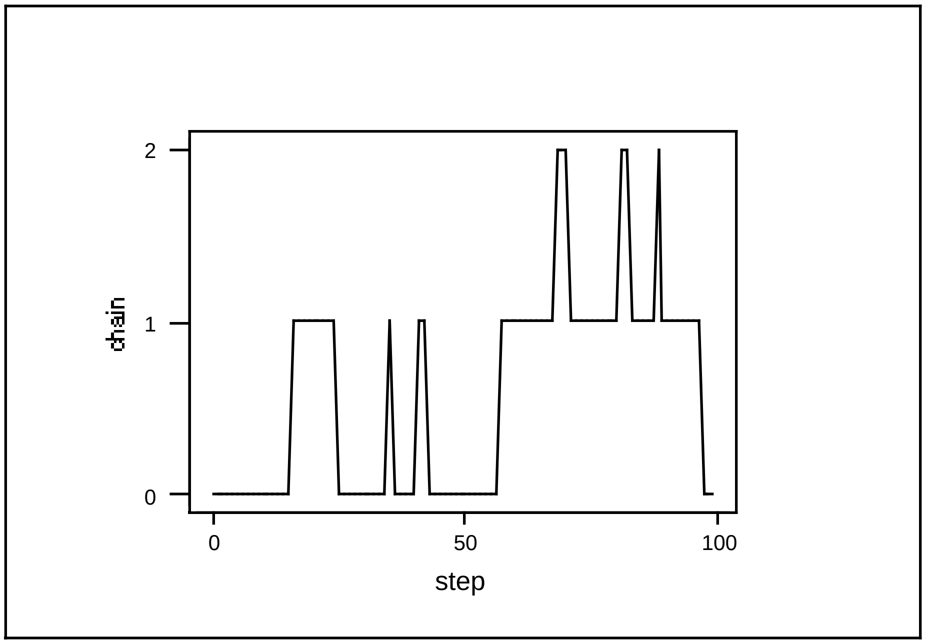
Recall that the Poisson Counting Process (same as Poisson Process) was the limit as the frame size $\Delta t \to 0$ of the Bernoulli Counting Process. It is also true that the M/M/1 Queuing Process is the limit as as the frame size $\Delta t \to 0$ of the Bernoulli Queuing Process.

For the M/M/1 queue, we have a continuous time arrival process (of customers, say) and a continuous time service distribution (of the time it takes to serve one customer).

Focus on service bay and queue there - arrivals at random times, and service time is less than interarrival time on average, but queue can still build up.

Example from last time (and text Example 7.4-1): Bernoulli Queuing Process Simulated as Markov Chain.

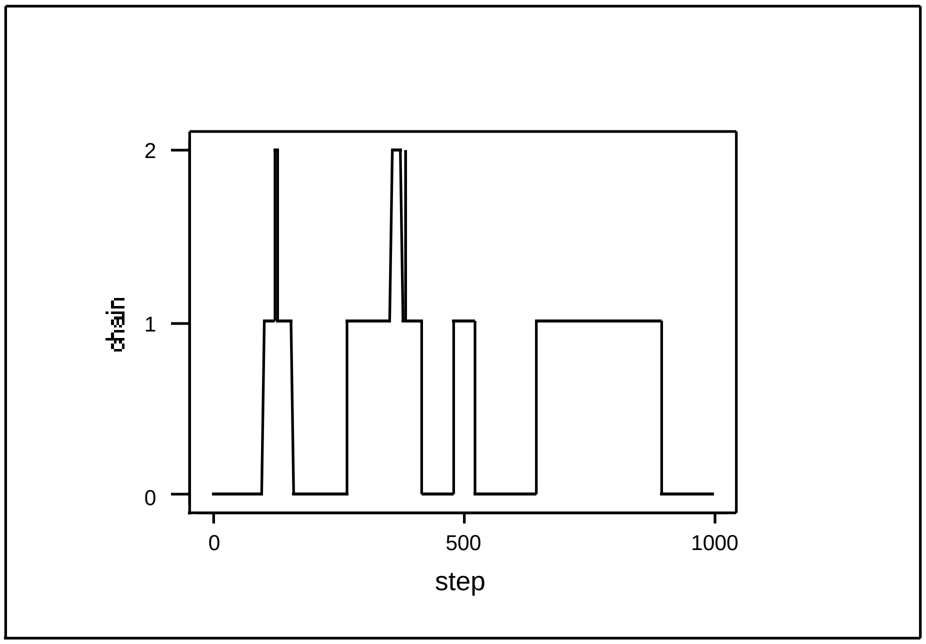
Arrival rate .10 per frame, service rate .15 per frame.



If I use ten times as many frames, I will be close to reproducing the Poisson queuing process.
The rates .10 and .15 become .010 and .015 in order to model the same interarrival and service distributions and this will imply a new transition matrix (with only 10% as many transitions per frame but 10 times more frames)

```
 0.990000  0.010000  0.000000  0.000000  0.000000  0.000000
 0.014850  0.975300  0.009850  0.000000  0.000000  0.000000
 0.000000  0.148500  0.975300  0.009850  0.000000  0.000000
 0.000000  0.000000  0.148500  0.975300  0.009850  0.000000
 0.000000  0.000000  0.000000  0.148500  0.975300  0.009850
 0.000000  0.000000  0.000000  0.000000  0.148500  0.985150
```

and the simulation produces a similar looking queue except the changes can be a bit more rapid occasionally.



But this is still only an approximation to the continuous time process. How can we simulate the continuous time process exactly?

Recall that inter-arrival times in a Poisson process are exponential with the same rate constant (rate at which events occur). To simulate a Poisson process, all we need is the interarrival times.

```
expos
  14.1513    8.0342    5.8156   28.7670    0.0012    1.9234    4.9803
   1.5266    3.5171    6.6002
```

Then cumulate these to make the event times:

```
           .     .   .                 :.  ..  .   .
        +---------+---------+---------+---------+---------+----- cumulated expos
        0        15        30        45        60        75
```

To construct the Poisson Process, which plots the event count against time, we need 21 sets of (x,y) points since
we need to start at (0,0) this is the 1

and
we need a horizontal line for each interarrival time:
We use the cumulated times to construct →

```
time
   0.0000   14.1513   14.1513   22.1855   22.1855   28.0011   28.0011
  56.7682   56.7682   56.7694   56.7694   58.6928   58.6928   63.6731
  63.6731   65.1997   65.1997   68.7168   68.7168   75.3171   75.3171
```

(doubling the cum times sets up the plotting abcissae)

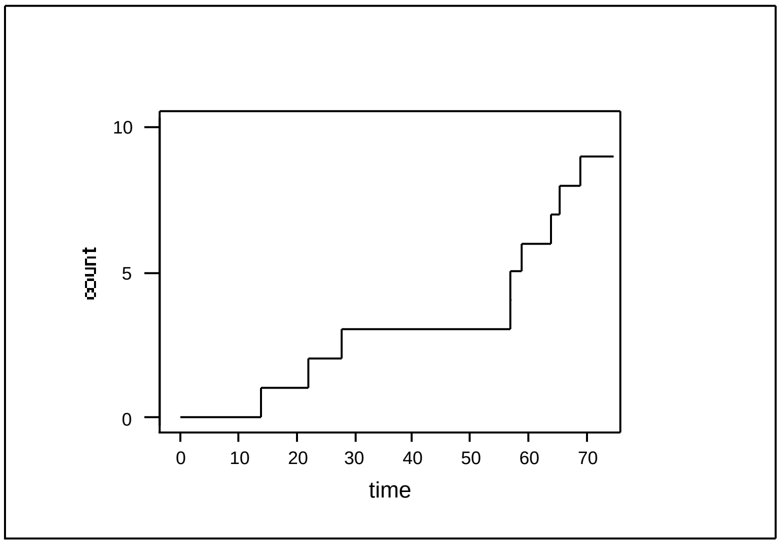
The ordinates of the plot will be the following sequence

```
count
    0     0     1     1     2     2     3     3     4     4     5     5     6
    6     7     7     8     8     9     9    10
```

So the points to plot are:

0.0000	0
14.1513	0
14.1513	1
22.1855	1
22.1855	2
28.0011	2
.	.
.	.
.	.
68.7168	9
75.3171	9
75.3171	10

and the graph of these points is



Important to note:

The exponential interarrival times were all that was needed to create the Poisson Counting Process outcome.

Can we do the same thing with the Poisson Queuing Process?

If we know all the interarrival times and service times, can we generate the Poisson Queuing Process?

For example, if the service times are

```
service
   0.7237    5.2724    4.8508   24.9249    2.3378    3.8762    0.5933
   2.5568   16.3412    1.6292
```

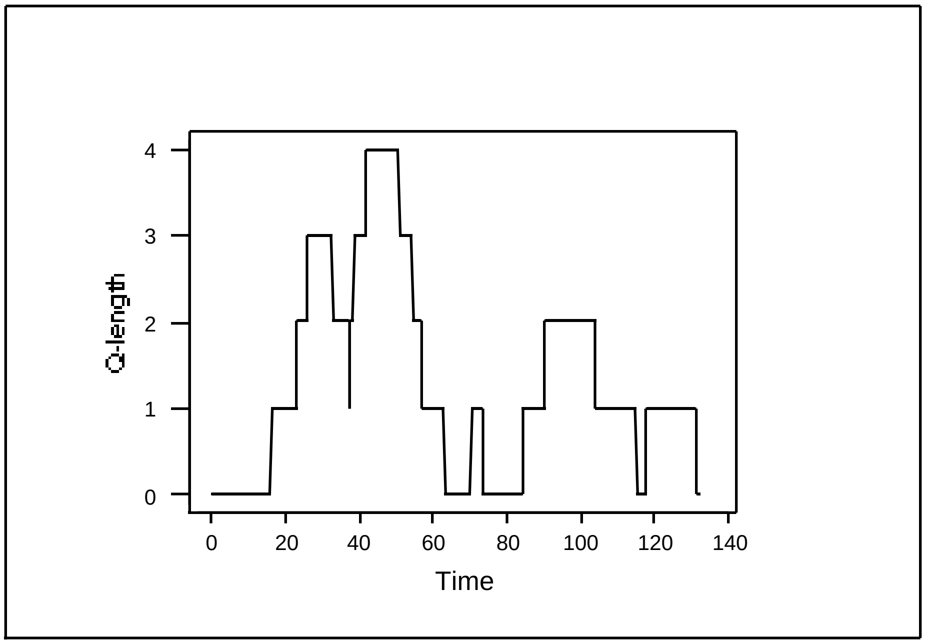
and the arrival times are (as in the Poisson example above)

```
arrival
  14.1513   22.1855   28.0011   56.7682   56.7694   58.6928   63.6731
  65.1997   68.7168   75.3171
```

Can we generate the queuing process.

The queue is 0 until 14.1513 then jumps to 1 but only until 14.1513+.7237 = 14.875 when it falls back to 0.
Then it stays at 0 until 22.1855 when it jumps to 1 again until 22.1855+5.2724 =27.4579
When it jumps back to 0 for a very short time jumping back up to 1 again at 28.0011.

And so on. It is a bit more complicated than a Poisson Process but can be programmed.



Here is the macro that does this (I have generated new arrivals and services for this graph):
(The program is described in the text on pages 320-322 - it works for k servers too)

```
Gmacro
MMk
erase c1-c13 k1-k14
let k1=10            # number of arrivals simulated = Nmax
rand k1 c1;           #simulate k1 interarrivals
expo 10.
rand k1 c2;           # simulate k1 service times
expo 6.67.
let k2=1              # number of servers
name c1 'arrivals' c2 'services'  k2 'no.servs'
name c3 'Tn' c4 'Dn' c5 'Fj' c6 'serverno' c7 'FJmin' c8 'Jmin' c9 'Bn' c10
'Ln' c11 'Wn'
let k3=1              #customer number
let c3(1)=0           # T(0)=c3(1)=0
let c4(1)=0           #D(0)=c4(1)=0
Do k4=1:k2      # c5 (j) is the dep time of cust most recently served by j
let c5(k4)=0              #F(j)=c5(j)  for j=1 to k2
let c6(k4)=k4             #servr numbers
enddo
let k5=k1+1
```

```
do k3=2:k5
let k6=k3-1
let c3(k3)=c3(k6)+c1(k6)   #Note c1(k6)=c1(k3-1)=A(n)
sort c5 c6 c7 c8
Let k8=c8(1)        # server with smallest completion time, Jmin
let k7=c7(1)        # smallest completion time , F(Jmin)
let k9=c3(k3)       #T(n)
rmax k7 k9 k10
let c9(k3)=k10       #B(n)
let c4(k3)=k10+c2(k6)   #D(n)
let c5(k8)=c4(k3)     # update F(Jmin)
let c10(k3)=c9(k3)-c3(k3)  #L(n)
let c11(k3)=c4(k3)-c3(k3)  #W(n)
enddo
print c1-c11
#           The following is to generate the time series of queue lengths.
max c4 k11            #how much time is needed on the horizontal axis
let k11=k11+1       # add a bit to make sure the graph shows return to 0
set c12
1:500
end
let c12=k11*c12/500  # define grid of time points at which queue size
#                                                                  computed
do k12=1:500             # for each grid point
let k14=0
do k13=2:k5              #scan all T(n) and D(n) to find queue
#                                             contributions (see p 320)
If ((c3(k13) le c12(k12)) and (c4(k13) gt c12(k12)))
      let k14=k14+1          # count if condition satisfied
endif
enddo
let c13(k12)=k14        # record queue length at this grid point
enddo
name c13 'Q-length' c12 'Time'
plot c13*c12;             # plot the simulated queue length time series
connect.
tally c13;
percent.
desc c13
endmacro
```

Simple in principle, but a bit complicated in detail!

The important thing is to understand the process of how the interarrival times and service times do produce the continuous time queue process.

Back to simpler stuff - finding the long run distribution of the queue length.

Could do by simulation of Markov Chain ….

Or mathematically…

Important idea: flow equation for queuing systems:

Each queue length is a 'state' of the queuing process.

If the service is fast enough to prevent an infinite queue build-up, there will be a long run distribution of queue length, and each state will be visited an infinite number of times.

Consider the transitions from queue length "1" to queue length "2". If the proportion of time the queue has length 1 is π_1, then the rate of leaving "1" for "2" is $\pi_1\lambda_A$. But this must equal the rate of arriving at "1" from "2", $\pi_2\lambda_{S.}$. In fact it is generally true that

$$\pi_i \lambda_A = \pi_{i+1} \lambda_S \text{ for } i= 0,1,2, \ldots$$

and this system of equations can be solved for the $\{\pi_i\}$ See p 295

$$\pi_i = (1-\lambda_A/\lambda_S)(\lambda_A/\lambda_S)^i \quad \text{for } i = 0,1,2, \ldots.$$

as long as $\lambda_A < \lambda_S$.

$r=\lambda_A/\lambda_S$ is called the arrival/service ratio.

In the M/M/1 queue,
we need r<1 if to avoid queue build-up and so we can consider the long run distribution of the queue length.
(In M/M/s for s>1, there is a similar but different condition).

Example: M/M/1 queue

r=.10/.15= 2/3 so π_i = (1-2/3)(2/3)^i for i = 0,1,2,…

Queue length	long run probability
0	0.33
1	0.22
2	0.15
3	0.10
4	0.066
	…….

Note: form of queue-length distribution is the similar to Geometric with p=1-r.

We know if X Geometric with p=1-r,
E(X) = (1-r)$^{-1}$

So $\sum_{i=1}^{\infty} i(r^{i-1})(1-r)$ = (1-r)$^{-1}$ and this can be used in computing
E(queue length) = $\sum_{i=0}^{\infty} ir^i(1-r)$ =r(1-r)$^{-1}$
Similarly SD (Queue Length) = $\sqrt{r}$ (1-r)$^{-1}$

What happens as r $\rightarrow$ 1?

Next: how long does a customer spend in the system?

Condition on queue length N at time customer arrives:

The wait will be N+1 service times, averaging (N+1) λ_S^{-1}

But $E(N) = r(1-r)^{-1}$ so expected time in system is

$$(r(1-r)^{-1} + 1)\,\lambda_S^{-1} = (1-r)^{-1}\,\lambda_S^{-1} = (\lambda_S - \lambda_A)^{-1}$$

Note what happens when λ_S not much bigger than λ_A.

Section 7.6 k-server Queuing Process

Reconsider how 2 servers would handle the services and arrivals we had before:

service

0.7 5.2 4.9 24.9 2.3 3.9 0.6 2.6 16.3 1.6

and the arrival times are (as in the Poisson example above)

arrival

14.1 22.2 28.0 56.8 56.9 58.7 63.7 65.2 68.7 75.3

Note that each arrival has a service time attached. There are the same number of service times as arrivals.

Consider the effect of 1 server:

Queue length sequence 01010101234567

2 servers:

Queue length sequence 01010101210101012

Much less queuing!

Again, Bernoulli queuing process is easy to construct, but MMk Continuous queue is more complicated, but possible to program.