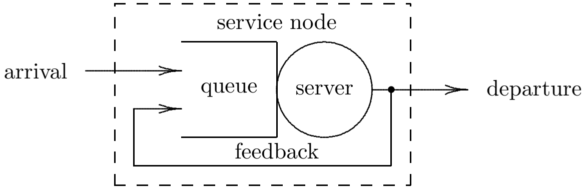
Section 3.3: Discrete-Event Simulation Examples

Discrete-Event Simulation: A First Course

©2006 Pearson Ed., Inc.   0-13-142917-5

SSQ with Immediate Feedback

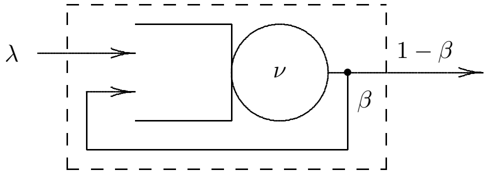
- If the service a job receives is incomplete or unsatisfactory, the job feeds back



- Completion of service and departure now have different meanings

Model Considerations

- When feedback occurs the job joins the queue consistent with the queue discipline
- The decision to depart or feed back is random with *feedback probability* β



  - λ is the arrival rate
  - ν is the service rate

Model Considerations

- Feedback is independent of past history
- In theory, a job may feed back arbitrarily many times
- Typically β is close to 0.0

GetFeedback Method

```
int GetFeedback(double beta)   /* 0.0 <= beta < 1.0 */
{
    SelectStream(2);
    if (Random() < beta)
        return (1);     /* feedback occurs */
    else
        return (0);     /* no feedback */
}
```

Statistical Considerations

- Index $i = 1, 2, 3, \ldots$ counts jobs that enter the service node
  - fed-back jobs are not recounted
- Using this indexing, all job-averaged statistics remain valid
  - We must update delay times, wait times, and service times *for each feed back* (The new service for the same job increases times but it must not cause an increase of #jobs)
- Jobs from outside the system are merged with jobs from the feedback process
- The steady-state request-for-service rate is larger than λ by the positive additive factor $\beta \bar{x} \nu$ [beta is the prob. of feedback, nu is the service rate, x is the time-avg no of jobs
- Note that $\bar{s}$ increases with feedback but $1/\nu$ is the average service time per request

Algorithm and Data Structure Considerations

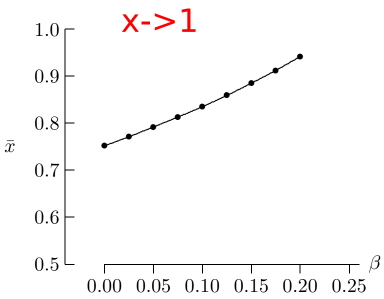
- **Example 3.3.1**

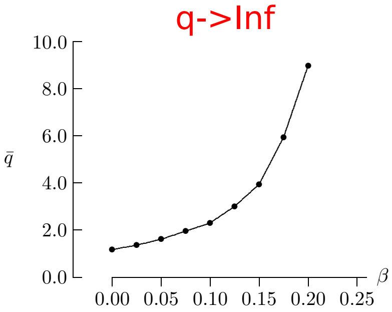
job index	1	2	3	4	5	·	6	·	7	8	·	9	···
arrival/ feedback	1	3	4	7	10	**13**	14	**15**	19	24	**26**	30	···
service	9	3	2	4	7	5	6	3	4	6	3	7	···
completion	10	**13**	**15**	19	**26**	31	37	**40**	44	50	53	**60**	···

- At the computational level, some algorithm and data structure is necessary

Example 3.3.2

- Program ssq2 was modified to incorporate immediate feedback
  - Interarrivals = *Exponential*(2.0)
    Service times = *Uniform*(1.0, 2.0)





- It appears saturation is achieved as $\beta \to 0.25$

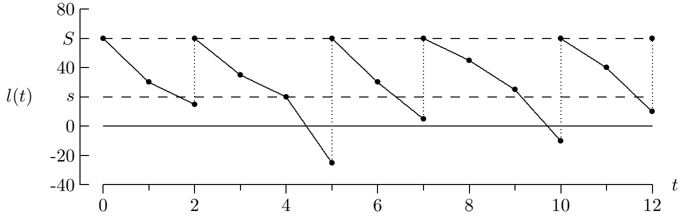
Flow Balance and Saturation

- Jobs flow into the service node at the average rate of λ
- To remain flow balanced jobs must flow out of the service node at the same average rate
- The average rate at which jobs flow out of the service node is $$\bar{x}(1-\beta)\nu$$
- Flow balance is achieved when $\lambda = \bar{x}(1-\beta)\nu$
- Saturation occurs when $\bar{x} = 1$ or as $\beta \to 1 - \lambda/\nu = 0.25$

(see prev. slide)

Simple Inventory System with Delivery Lag

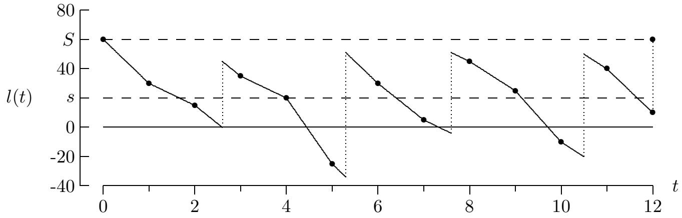
- *Delivery lag* or *lead time* occurs when orders are not delivered immediately
- Lag is assumed to be random and independent of order size
- Without lag, inventory jumps occur only at inventory review times



without delivery lag: discontinuity when orders to suppliers occurs to re-fill to the value s

SIS with Lag

- With delivery lag, inventory jumps occur at arbitrary times

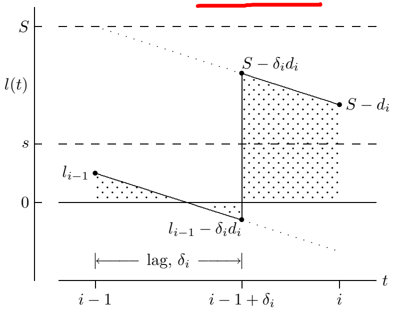


- The last order is assumed to have no lag
- We assume that orders are delivered before the next inventory review
- With this assumption, there is no change to the specification model

Statistical Considerations

- If $l_{i-1} \geq s$ the equations for $\bar{l}_i^+$ and $\bar{l}_i^-$ remain correct
- When delivery lag occurs the time-averaged holding and shortage intervals must be modified
  - The delivery lag for interval i is $\underline{0 < \delta_i < 1}$

l+ = time-averaged holding value
l- = time-averaged shortage level



The inventory level drop at constant rate
to [l_(i-1) - delta_i * d_i], where d_i is the demand at time i (note that delta_i < 1)

The area of each figures must be determined differently.

Consistency Checks

- It is fundamentally important to verify extended models with the parent model
  - Set system parameters <u>to special values</u>
- Set $\underline{\beta = 0}$ for the SSQ with feedback [same results of SSQ without feedback]
  - Verify that all statistics <u>agree</u> with <u>parent</u>
- Using the library `rngs` facilitates this kind of comparison
- It is a good practice to check for intuitive "small-perturbation" consistency
  - Use a small, but non-zero β and check that appropriate statistics are slightly larger

Example 3.3.3

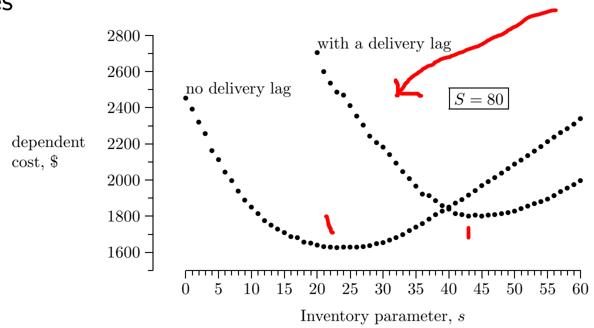
- For the SIS with delivery lag, $\delta_i = 0.0$ iff no order during $i^{\rm th}$ interval, $0 < \delta_i < 1.0$ otherwise
- The SIS is *lag-free* iff $\delta_i = 0.0$ for all i
- If (S, s) are fixed then, even with small delivery lags:
  - $\bar{o}$, $\bar{d}$, and $\bar{u}$ are the same regardless of delivery lag
  - Compared to the lag-free system, $\bar{l}^+$ will decrease
  - Compared to the lag-free system, $\bar{l}^-$ will increase or remain unchanged

o = average orders
d = average demands
u = order frequency = (no. orders / n)

Example 3.3.4

sis3.c

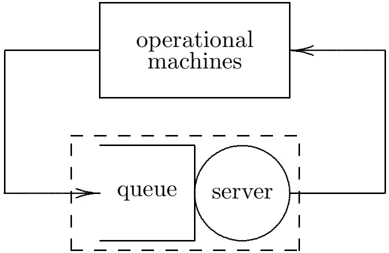
- Delivery lags are independent *Uniform*(0.0, 1.0) random variates



- Delivery lag causes $\bar{l}^+$ to decrease and $\bar{l}^-$ to increase or remain the same
- $C_{\text{hold}} = \$25$ and $C_{\text{short}} = \$700$ cause shift up and to the left

Single-Server Machine Shop

- The machine shop model is closed because there are a finite number of machines in the system



  - Assume repair times are *Uniform*(1.0, 2.0) random variates
  - There are *M* machines that fail after an *Exponential*(100.0) random variate

Program ssms

Two random streams:
- repair times
- failures

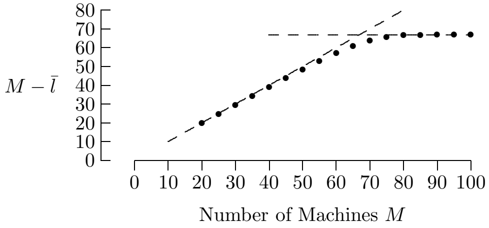
Similar to ssq2, but..

- Program `ssms` simulates a single-server machine shop
- The library `rngs` is used to uncouple the random processes
- The failure process is defined by the array `failures`
  - A $\mathcal{O}(M)$ search is used to find the next failure
  - Alternate data structures can be used to increase computational efficiency

array of failure times

Example 3.3.5

- The time-averaged number of working machines is $M - \bar{l}$



- For small values of M the time-averaged number of operational machines is essentially M
- For large values of M this value is essentially constant at approximately 67 [!!]