Discrete-Event Simulation:
A First Course

Section 3.3: Discrete-Event Simulation Examples
Outline

- Single-server service node with immediate feedback
- A simple inventory system with delivery lag
- A single-server machine shop
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 feedback is random with feedback probability $\beta$.

\[ \lambda, \nu \]

- $\lambda$ is the arrival rate.
- $\nu$ is the service rate.
Model Considerations

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

GetFeedback Method

```c
int GetFeedback(double beta) /* 0.0 <= beta < 1.0 */
{
    SelectStream(2);
    if (Random() < beta)
        return (1); /* feedback occurs */
    else
        return (0); /* no feedback */
}
```
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 feedback
- Jobs from outside the system are merged with jobs from the feedback process
- The steady-state request-for-service rate is larger than $\lambda$ by the positive additive factor $\beta \bar{x} \nu$
- Note that $\bar{s}$ increases with feedback but $1/\nu$ is the average service time per request
Example 3.3.1

<table>
<thead>
<tr>
<th>job index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrival/</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>19</td>
<td>24</td>
</tr>
<tr>
<td>feedback</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>service</td>
<td>10</td>
<td>13</td>
<td>15</td>
<td>19</td>
<td>26</td>
<td>31</td>
<td>37</td>
<td>40</td>
<td>44</td>
<td>50</td>
</tr>
<tr>
<td>completion</td>
<td>10</td>
<td>13</td>
<td>15</td>
<td>19</td>
<td>26</td>
<td>31</td>
<td>37</td>
<td>40</td>
<td>44</td>
<td>50</td>
</tr>
</tbody>
</table>

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 $\beta \rightarrow 0.25$. 
Flow Balance and Saturation

- Jobs flow into the service node at the average rate of $\lambda$.
- 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 $\beta \to 1 - \lambda/\nu = 0.25$. 
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.
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 $0 < \delta_i < 1$. 

[Diagram showing inventory levels and delivery lag with equations and intervals labeled.]
Consistency Checks

- It is fundamentally important to verify extended models with the parent model
  - Set system parameters to special values
- Set $\beta = 0$ for the SSQ with feedback
  - Verify that all statistics agree with parent
- 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 $\beta$ 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 $i^{th}$ 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, $\bar{T}^+$ will decrease
  - Compared to the lag-free system, $\bar{T}^-$ will increase or remain unchanged
Example 3.3.4

- 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
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

- 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 $O(M)$ search is used to find the next failure
  - Alternate data structures can be used to increase computational efficiency
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