# Computer Systems Performance Analysis and Benchmarking (37-235)

### Analytic Modeling
### Simulation
### Measurements / Benchmarking

### Lecture/Assignments/Projects:
Dr. Christian Kurmann

### Textbook:
Raj Jain, “The Art of Computer Systems Performance Analysis”, 1991 Wiley & Sons, New York

### Topic of Today:
- **Verification / Validation**
- **Transient Removal**
- **Random Generators for Simulation**



# Verification vs. Validation

### Verification of a simulator
- Is the simulator correct?

### Validation of a Simulator
- Is the simulator applicable?
- Are the assumptions reasonable?

### Simulators are complex pieces of software:
- Modularity
  - some generic modules
  - some domain dependent modules
  - some model specific modules



# Example

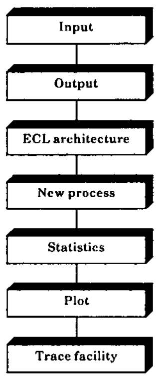



**FIGURE 25.1** Layered structure of the congestion simulation model.

**Example 25.1** Figure 25.1 shows the modules for a computer network simulation developed for congestion control studies. The model simulates a network with a number of source nodes, a number of intermediate nodes, and a number of destinations, as shown in Figure 25.2. Packets start from the source nodes, travel through a number of prespecified intermediate nodes (called paths), and reach the destination. The packet sizes and service times at various nodes are randomly distributed. In Figure 25.2, $S_i$'s are sources, $R_i$'s are intermediate nodes, and $D_i$'s are destinations. The model simulates $n$ sources sharing a common path throught $m$ intermediate nodes for any given $n$ and $m$. This is equivalent to two local-area networks connected through $m$ intermediate nodes.



# Example

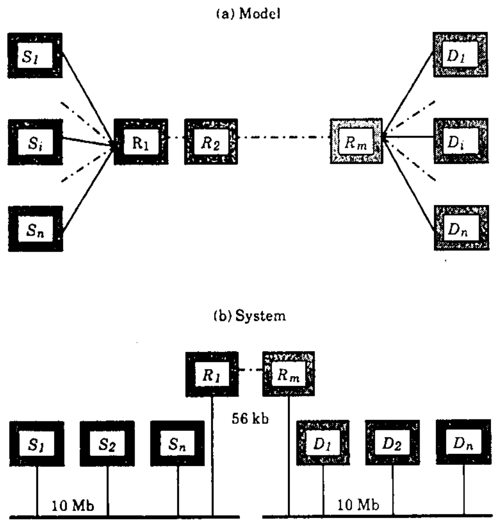



**FIGURE 25.2** Model of two interconnected local-area networks.

## Techniques:

- The usual game in software engineering...
- Modular design
- Debugging
- Structured walk-through
- Deterministic models
- Simplified cases
- On-line graphic animation
- Traces
- Continuity tests
- Degeneracy tests
- Consistency tests
- Seed independence tests

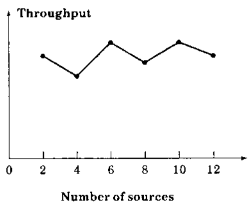


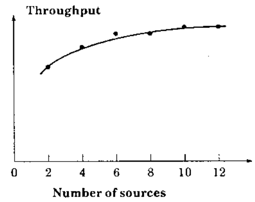
## Traces/Continuity tests

PKT=100 CR=4 #S=2 RS=2 — ! This was a run with 100 packets, 4 credits per source, two sources. Router service time was twice that of the source.

| Time | Node: Event | Pkt #/ Attempt | Sample Delay | Delay Estimate | |
|---|---|---|---|---|---|
| 0.00: | S1: TIMR | 1- 1 | | | ! Round trip delay-measuring stop watch started by source 1 |
| | S1: SEND | 1- 1/ 1 | | | ! Packet 1 sent by source 1 |
| | S1: STRT | | | | ! Timeout alarm clock set by source 1 |
| | S2: TIMR | 2- 1 | | | ! Round trip delay-measuring stop watch started by source 2 |
| | S2: SEND | 2- 1/ 1 | | | ! Packet 1 sent by source 2 |
| | S2: STRT | | | | ! Timeout alarm clock set by source 2 |
| 1.00: | R1: QUED | 1- 1/ 1 | | | ! Packet 1 of source 1 was put into a queue at router 1 |
| | R1: LOST! | 2- 1/ 1 | | | ! Packet 1 of source 2 was lost due to lack of buffer at router 1 |
| 3.00: | D1: RECD | 1- 1/ 1 | | | ! Packet 1 of source 1 was received at destination 1 |
| | S1: ACKD | 1- 1 | | | ! Acknowledgment for packet 1 was received at source 1 |
| | S1: UPDT | 1- 1 | 3.00 | 3.00 | ! Source 1 updated its estimate of round trip delay |

**FIGURE 25.3** Sample packet event-trace output from the simulation model.






## Validation Techniques

### Three Key Aspects of the system

- Assumptions
- Input parameter values
- output parameter values - conclusions

### Methods

- Expert intuition
- Real systems measurements
- Theoretical results

### Expert Intuition Example

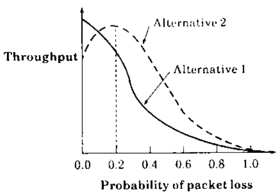



**FIGURE 25.6** Example of problems caused by invalid assumptions that are easily detected by experts.



## Simple Statistical Methods for...

- Transient removal
- Termination tail removal
- Stopping criteria
- Regenerative systems

### Transient removal

- Long runs
- Proper initialization
- Truncation
- Initial data deletion
- Moving average of independent replications
- Batch means

# Truncation

1,2,3,4,5,6,7,8,9,10,11,10,9,10,11,10,9,10

**Example 25.2** Consider the following sequence of observations: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 10, 9, 10, 11, 10, 9, 10, 11, 10, 9,....

Ignoring the first observation ($l = 1$), the range of the remaining observations is (2, 11). Since the second observation is equal to the minimum, the transient phase is longer than 1.

Ignoring the first two observation ($l = 2$), the range of the remaining sequence is (3, 11). Again, the next (third) observation is equal to the minimum; the truncation continues with $l = 3$ and so on.

Finally, at $l = 9$ the range of the remaining sequence is (9, 11), and the tenth observation 10 is neither the minimum nor the maximum. The length of the transient interval is therefore 9, and the first nine observations are discarded.

A trajectory of this set of observations is shown in Figure 25.7. It is seen from the figure that the transient phase for this data does indeed end after nine observations. □

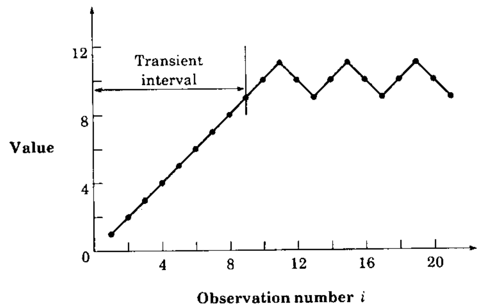



**FIGURE 25.7** Plot of the data used in the truncation method example.



# Systematic Method

## Initial Data Deletion

1. Get a mean trajectory by averaging across replications:

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}, \qquad j = 1,2,\ldots,n$$

Figure 25.8a shows the trajectories of several replications, and an average trajectory is shown in Figure 25.8b.

2. Get the overall mean:

$$\bar{\bar{x}} = \frac{1}{n}\sum_{j=1}^{n} \bar{x}_j$$

Set $l = 1$ and proceed to the next step.

3. Assuming that the transient state is only $l$ long, delete the first $l$ observations from the mean trajectory and get an overall mean from the remaining $n - l$ values:

$$\bar{\bar{x}}_l = \frac{1}{n-l}\sum_{j=l+1}^{n} \bar{x}_j$$

4. Compute the relative change in the overall mean:

$$\text{Relative change} = \frac{\bar{\bar{x}}_l - \bar{\bar{x}}}{\bar{\bar{x}}}$$

5. Repeat steps 3 and 4 by varying $l$ from 1 to $n - 1$. Plots of the overall mean and the relative change as functions of $l$ are shown in Figure 25.8c and 25.8d. After a certain value of $l$, the relative change graph stabilizes. This point is known as the knee, and the value of $l$ at the knee is the length of the transient interval.



# Moving Average of Independent Replic.

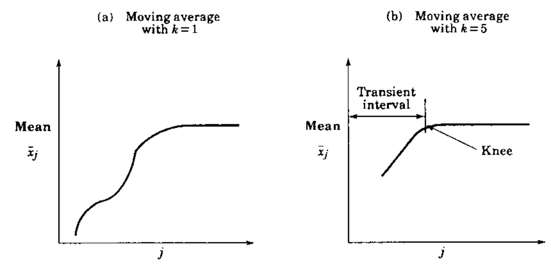



**FIGURE 25.9** Moving average of independent replications.

1. Get a mean trajectory by averaging across replications:

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}, \qquad j = 1,2,\ldots,n$$

Set $k = 1$ and proceed to the next step.

2. Plot a trajectory of the moving average of successive $2k + 1$ values:

$$\bar{\bar{x}}_j = \frac{1}{2k+1}\sum_{l=-k}^{k} \bar{x}_{j+l}, \qquad j = k+1, k+2, \ldots, n-k$$

3. Repeat step 2, with $k = 2, 3, \ldots$ until the plot is sufficiently smooth.
4. Find the knee of the plot. The value of $j$ at the knee gives the length of the transient phase.



# Batch Means

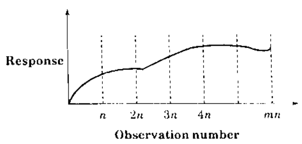



**FIGURE 25.10** Transient removal by batch means requires dividing the data into $m$ batches of size $n$ each.

1. For each batch, compute a batch mean:

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \qquad i = 1,2,\ldots m$$

2. Compute the overall mean:

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m} \bar{x}_i$$

3. Compute the variance of the batch means:

$$\text{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m} \left(\bar{x}_i - \bar{\bar{x}}\right)^2$$

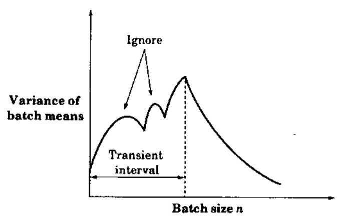



**FIGURE 25.11** Transient removal by batch means.

## Stopping Criteria / Variance Estimation

## Independent Replication

1. Compute a mean for each replication:

$$\bar{x}_i = \frac{1}{n}\sum_{j=n_0+1}^{n_0+n} x_{ij}, \qquad i = 1,2,\ldots,m$$

2. Compute an overall mean for all replications:

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m}\bar{x}_i$$

3. Calculate the variance of replicate means:

$$\text{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

The confidence interval for the mean response is*

$$[\bar{\bar{x}} \mp z_{1-\alpha/2}\text{Var}(\bar{x})]$$

## Batch means

**TABLE 25.1 Autocovariance and Variance for Various Batch Sizes**

| Batch Size | Autocovariance | Variance |
|---|---|---|
| 1 | −0.18792 | 1.79989 |
| 2 | 0.02643 | 0.81173 |
| 4 | 0.11024 | 0.42003 |
| 8 | 0.08979 | 0.26437 |
| 16 | 0.04001 | 0.17650 |
| 32 | 0.01108 | 0.10833 |
| 64 | 0.00010 | 0.06066 |
| 128 | −0.00378 | 0.02992 |
| 256 | 0.00027 | 0.01133 |
| 512 | 0.00069 | 0.00503 |
| 1024 | 0.00078 | 0.00202 |



## Batch Means

1. Compute means for each batch:

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \qquad i = 1,2,\ldots,m$$

2. Compute an overall mean:

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m}\bar{x}_i$$

3. Calculate the variance of batch means:

$$\text{Var}(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

The confidence interval for the mean response is

$$[\bar{\bar{x}} \mp z_{1-\alpha/2}\text{Var}(\bar{x})]$$

$$\text{Cov}(\bar{x}_i, \bar{x}_{i+1}) = \frac{1}{m-2}\sum_{i=1}^{m-1}(\bar{x}_i - \bar{\bar{x}})(\bar{x}_{i+1} - \bar{\bar{x}})$$

This quantity is also called the **autocovariance.**

## Method of Regeneration

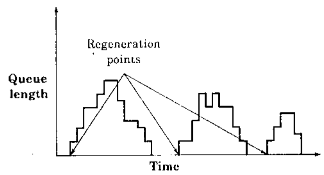



**FIGURE 25.13** Regeneration points.



Suppose you have a regenerative simulation consisting of $m$ cycles of sizes $n_1, n_2, \ldots, n_m$, respectively. Cycle means are given by

$$\bar{x}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij}$$

However, the overall mean is not an arithmetic mean of cycle means:

$$\bar{\bar{x}} \neq \frac{1}{m}\sum_{i=1}^{m}\bar{x}_i$$

The correct procedure to compute the overall mean and its confidence interval is as follows:

1. Compute cycle sums:

$$y_i = \sum_{j=1}^{n_i} x_{ij}$$

2. Compute the overall mean:

$$\bar{\bar{x}} = \frac{\sum_{i=1}^{m} y_i}{\sum_{i=1}^{m} n_i}$$

3. Calculate the difference between expected and observed cycle sums:

$$w_i = y_i - n_i\bar{\bar{x}}, \qquad i = 1,2,\ldots,m$$

4. Calculate the variance of the differences:

$$\text{Var}(w) = s_w^2 = \frac{1}{m-1}\sum_{i=1}^{m} w_i^2$$

5. Compute the mean cycle length:

$$\bar{n} = \frac{1}{m}\sum_{i=1}^{m} n_i$$

The confidence interval for the mean response is given by

$$\bar{\bar{x}} \mp z_{1-\alpha/2}\frac{s_w}{\bar{n}\sqrt{m}}$$



## Random Generators

- Used for simulators
- True randomness is not desired (reproducability for debugging)
- Statistical properties must be random
- No cryptographic strength required

# Example

$$x_n = f(x_{n-1}, x_{n-2}, \ldots)$$

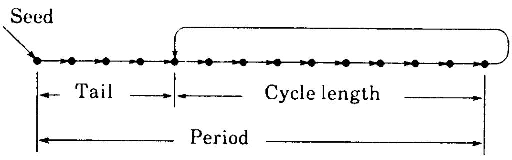
One such function is

$$x_n = 5x_{n-1} + 1 \bmod 16$$

$$x_1 = 5(5) + 1 \bmod 16 = 26 \bmod 16 = 10$$

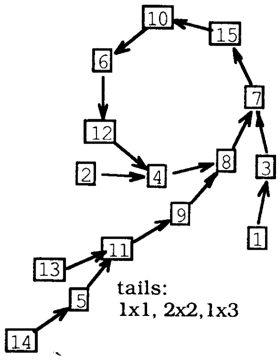
10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7,
4, 5, 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9,

**cycle length** **seed.**






# Cycles






# Properties

1. *It should be efficiently computable.* Since simulations typically require several thousand random numbers in each run, the processor time required to generate these numbers should be small.
2. *The period should be large.* A small period may cause the random-number sequence to recycle, resulting in a repeated event sequence. This may limit the useful length of simulation runs.
3. *The successive values should be independent and uniformly distributed.* The correlation between successive numbers should be small. Correlation, if significant, indicates dependence.

- Linear-congruential generators
- Tausworthe generators
- Extended Fibonacci generators
- Combined generators



# Linerar Congruence Generators

$$x_n = a^n \bmod m$$

$$x_n = ax_{n-1} \bmod m$$

$$x_n = ax_{n-1} + b \bmod m$$

$$a = 23 \text{ and } m = 10^8 + 1.$$

**full-period generator.**

lower autocorrelation

$$x_n = (2^{34} + 1)x_{n-1} + 1 \bmod 2^{35}$$

$$x_n = (2^{18} + 1)x_{n-1} + 1 \bmod 2^{35}$$

**Multiplicative LCG**

$$x_n = ax_{n-1} \bmod m$$

$$m = 2^k$$

$$x_n = 5x_{n-1} \bmod 2^5$$

$$x_n = 7x_{n-1} \bmod 2^5$$

# Examples

$$m \neq 2^k$$

$$x_n = 3x_{n-1} \bmod 31$$

$$5^3 \bmod 31 = 125 \bmod 31 = 1$$

$$x_n = 7^5 x_{n-1} \bmod (2^{31} - 1)$$



# Shiftregister Generators

**TAUSWORTHE GENERATORS**

$$b_n = c_{q-1}b_{n-1} \oplus c_{q-2}b_{n-2} \oplus c_{q-3}b_{n-3} \oplus \cdots \oplus c_0 b_{n-q}$$

$$x^7 + x^3 + 1$$

$$D^7 b(n) + D^3 b(n) + b(n) = 0 \bmod 2$$

$$b_{n+7} + b_{n+3} + b_n = 0 \bmod 2,$$

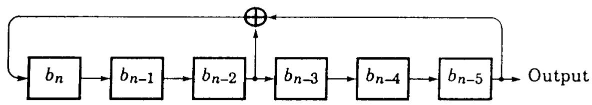
$$b_7 = b_3 \oplus b_0 = 1 \oplus 1 = 0$$

$$b_8 = b_4 \oplus b_1 = 1 \oplus 1 = 0$$

$$b_9 = b_5 \oplus b_2 = 1 \oplus 1 = 0$$

$$b_{10} = b_6 \oplus b_3 = 1 \oplus 1 = 0$$

$$b_{11} = b_7 \oplus b_4 = 0 \oplus 1 = 1$$




1111111 0000111 0111100 1011001 0010000
0010001 0011000 1011101 0110110
0000110 0110101 0011100 1111011
0100001 0101011 1110100 1010001
1011100 0111111 1000011 1000000.

period of 127.



# Chi-Square Test

$$D = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

identically distributed (**IID**) $U(0, 1)$,

testing random-variate generators.

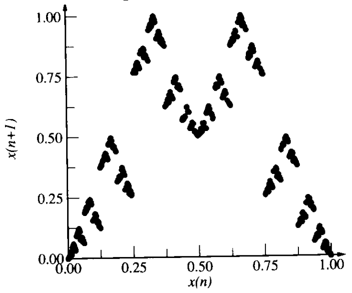
$D$ has a chi-square distribution

level of significance $\alpha$

computed $D$ is *less* than the $\chi^2_{[1-\alpha;k-1]}$

read from Table A.5

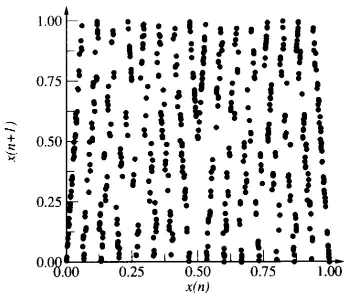
# Graphical Tests



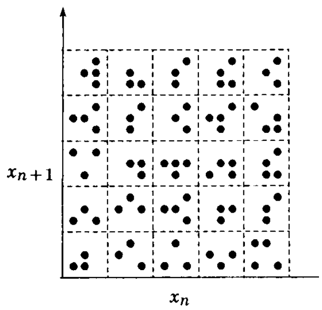

Plot of overlapping pairs
Tausworthe generator $x^{15} + x + 1$.



# Graphical Tests




Plot of overlapping pairs
Tausworthe generator $x^{15} + x^4 + 1$.




Two-dimensional uniformity