King Fahd University of Petroleum & Minerals
Computer Engineering Dept

COE 402 – Computer Systems Performance Evaluation

Term 043

Dr. Ashraf S. Hasan Mahmoud

Rm 22-148-3

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa

Simulations: Key Questions

- **Slides are by Raj 1994.**

Simulations: Key Questions

- What are the common mistakes in simulation and why most simulations fail
- What language should be used for developing a simulation model
- How to schedule events in a simulation?
- How to verify and validate a model?
- How to determine that the simulation has reached a steady state?
- How long to run a simulation

Simulations: Key Questions

- How to generate uniform random
- Numbers?
- How to verify that a given random number generator is good?
- How to select seeds for a random number generators?
- How to generate random variables with a given distribution?
- What distributions should be used and when?

Common Mistakes in Simulation

1. **Inappropriate Level of Detail More detail**
   - **More time → More Bugs → More CPU → More parameters ≠ More accurate**
2. **Improper Language**
   - **General purpose: More portable, More efficient, More time**
3. **Unverified Models: Bugs**
4. **Invalid Models: Model vs reality**
5. **Improperly Handled Initial Conditions**
6. **Too Short Simulations**
   - **Need confidence intervals**
7. **Poor Random Number Generators:**
   - **Safer to use a well known generator**
8. **Improper Selection of Seeds:**
   - **Zero seeds Same seeds for all streams**

Other Causes of Simulation Analysis Failure

1. **Inadequate Time Estimate**
2. **No Achievable Goal**
3. **Incomplete Mix of Essential Skills**
   - **Project Leadership**
   - **Modeling and Statistics**
   - **Programming**
   - **Knowledge of the Modeled System**
4. **Inadequate Level of User Participation**
5. **Obsolete or Nonexistent Documentation**
6. **Inability to Manage the Development of a Large Complex Computer Program Need software engineering tools**
7. **Mysterious Results**

Checklist for Simulations

1. Checks before developing a simulation
   - Is the goal of the simulation properly specified?
   - Is the level of detail in the model appropriate for the goal?
   - Does the simulation team include personnel with project leadership modeling programming and computer systems backgrounds?
   - Has sufficient time been planned for the project?
2. Checks during development
   - Has the random number generator used in the simulation been tested for uniformity and independence?
   - Is the model reviewed regularly with the end user?
   - Is the model documented?
3. Checks after the simulation is running
   - Is the simulation length appropriate?
   - Are the initial transients removed before computation?
   - Has the model been verified thoroughly?
   - Has the model been validated before using its results?
   - If there are any surprising results, have they been validated?
   - Are all seeds such that the random number streams will not overlap?

Terminology

- State Variables
  - Define the state of the system
  - Can restart simulation from state variables
  - E.g. length of the job queue
- Event
  - Change in the system state
  - E.g. arrival
  - beginning of a new execution
  - Departure
- Types of Models
  - Continuous Time and Discrete Time Models
  - CPU scheduling model vs number of students attending the class

Terminology – cont'd

- **Continuous-State and Discrete-State Models**
  - **Example: Time spent by students in a weekly class vs Number of students**
  - **Discrete-state = Discrete-event model**
  - **Continuous-state = Continuous-event model**
- **Continuity of time ≠ Continuity of state**
- **Four possible combinations**
  - **Discrete-state discrete-time**
  - **Discrete-state continuous-time**
  - **Continuous-state discrete-time**
  - **Continuous-state continuous-time models**

Terminology – cont'd

- **Deterministic and Probabilistic Models**
- **Static and Dynamic Models**
  - **CPU scheduling model vs $E = mc^2$**
- **Linear and Nonlinear Models**
  - **Output as a function of Input**
- **Open and Closed Models**
  - **E.g. External input → open models**
- **Stable and Unstable Models**
  - **Stable → Settles to steady state**
  - **Unstable → Continuously changing**

Terminology – cont'd

- **What kind of systems we will be interested in?**
- **Computer system models**
  - **Continuous time**
  - **Discrete state**
  - **Probabilistic**
  - **Dynamic**
  - **Nonlinear**
  - **Open or closed**
  - **Stable or unstable**

Selecting a Language for Simulation

1. **Simulation language**
2. **General purpose**
3. **Extension of a general purpose language**
4. **Simulation package**

Simulation languages

- Save development time
- Built in facilities for time advancing event scheduling entity manipulation random variate generation statistical data collection and report generation
- More time for system specification issues
- Very readable modular code

General purpose language

- Analyst s familiarity
- Easy availability
- Quick startup
- Time for routines for event handling random number generation
- Other Issues: Efficiency, flexibility and portability
- Recommendation: Learn at least one simulation language

Types of Simulations

1. **Emulation Using hardware or firmware**
   - **E.g. Terminal emulator, processor emulator**
   - **Mostly hardware design issues**
2. **Monte Carlo Simulation**
3. **Trace Driven Simulation**
4. **Discrete Event Simulation**

Monte Carlo Simulation

- **Static simulation (No time axis)**
- **To model probabilistic phenomenon**
- **Need pseudorandom numbers**
- **Used for evaluating nonprobabilistic expressions using probabilistic methods**

Example: Monte Carlo Simulation

- **Problem: Evaluate** $I = \int_0^2 e^{-x^2}\,dx$
- **Solution: Let *X* be ~U(0,2) ➔ $f_X(x) = ½$ 0≤*x*≤2.**

define $y = 2e^{-x^2}$

It is obvious that the mean value of y is obtained using $\bar{y} = \int_0^2 2e^{-x^2} f(x)\,dx = \int_0^2 e^{-x^2}\,dx$

i.e. the mean value of y is the integral I.

Therefore, we can estimate y using $\bar{y} \approx \frac{1}{n}\sum_{i=1}^{n} 2e^{-x_i^2}$

Example: Monte Carlo Simulation – cont'd

- **Solution-cont'd: As the number of samples n increases, the more accurate the estimate of the mean or integral value.**

n	Mean of y
10	1.7764
100	1.4579
1000	1.4995
100000	1.4933
1000000	1.4937

Trace Driven Simulation

- **Trace = time ordered record of events on a system**
- **Trace-driven simulation = trace input**
- **Used in analyzing or tuning resource management algorithms:**
  - **Paging, cache analysis, CPU scheduling, deadlock prevention, dynamic storage allocation**
- **Example: Trace = Page reference patterns**
- **Should be independent of the system under study**
  - **E.g. trace of pages fetched depends upon the working set size and page replacement policy**
  - **Not good for studying other page replacement policies**
  - **Better to use pages referenced**

Advantages of Trace-Driven Simulations

1. **Credibility**
2. **Easy Validation: Compare simulation with measured**
3. **Accurate Workload: Models correlation and interference**
4. **Detailed Trade Offs: Detailed workload ➔ Can study small changes in algorithms**
5. **Less Randomness: Trace ➔ deterministic input ➔ Fewer repetitions**
6. **Fair Comparison: Better than random input**
7. **Similarity to the Actual Implementation: Trace driven model is similar to the system**
   - **Can understand complexity of implementation**

Disadvantages of Trace-Driven Simulations

1. Complexity: More detailed
2. Representativeness: Workload changes with time, equipment
3. Finiteness: Few minutes fill up a disk
4. Single Point of Validation: One trace one point
5. Detail
6. Trade-Offs: Difficult to change workload

Discrete Event Simulations

- Concentration of a chemical substance ➔ Continuous event simulations
- Number of jobs Discrete event ➔ Discrete state discrete time

Components of Discrete Event Simulations

1. **Event Scheduler**
    - Schedule event X at time T
    - Hold event X for a time interval dt
    - Cancel a previously scheduled event X
    - Hold event X indefinitely
    - Schedule an indefinitely held event
2. **Simulation Clock and a Time Advancing Mechanism**
    - unit time approach
    - event driven approach
3. **System State Variables**
    - Global → Number of jobs
    - Local → CPU time required for a job
4. **Event Routines One per event**
    - E.g. job arrivals, job scheduling, and job departure

Components of Discrete Event Simulations – cont'd

5. **Input Routines: Get model parameters, Vary parameters in a range**
6. **Report Generator**
7. **Initialization Routines: Set the initial state, Initialize seeds**
8. **Trace Routines: On/off feature**
9. **Dynamic Memory Management: Garbage collection**
10. **Main Program**

Event Set Algorithms

Analysis of Simulation Results

1. **Analysis of Simulation Results**
2. **Model Verification Techniques**
3. **Model Validation Techniques**
4. **Transient Removal**
5. **Terminating Simulations**
6. **Stopping Criteria Variance Estimation**
7. **Variance Reduction**

Model Verification vs Validation

- Verification ➔ Debugging
- Validation ➔ Model = Real world
- Four Possibilities
  - Unverified, Invalid
  - Unverified, Valid
  - Verified, Invalid
  - Verified, Valid

Model Verification Techniques

1. Top Down Modular Design
2. Antibugging
3. Structured Walk Through
4. Deterministic Models
5. Run Simplified Cases
6. Trace
7. On Line Graphic Displays
8. Continuity Test
9. Degeneracy Tests
10. Consistency Tests
11. Seed Independence

Top Down Modular Design

- **Divide and Conquer**
- **Modules: Subroutines, Subprograms, Procedures**
  - **Modules have well defined interfaces**
  - **Can be independently developed, debugged, and maintained**
- **Top-down design → hierarchical structure → Modules and submodules**

Verification Techniques

- **Antibugging: Include self-checks**
  - **E.g. Σ Probabilities = 1**
  - **Jobs left = Generated - Serviced**
- **Structured Walk Through**
  - **Explain the code another person or group**
  - **Works even if the person is sleeping**
- **Deterministic Models: Use constant values**
- **Run Simplified Cases:**
  - **Only one packet**
  - **Only one source**
  - **Only one intermediate node**

Trace

- Trace: Time ordered list of events and variables
- Several levels of detail:
    - Events trace
    - Procedure trace
    - Variables trace
- User selects the detail
- Include on and off switch
    - Remember the boolean DebugFlag variable

On-Line Graphic Displays

- Make simulation interesting
- Help selling the results
- More comprehensive than trace

Continuity Test

- Run for different values of input parameters
- Slight change in input ➔ (usually) slight change in output

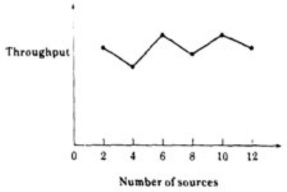


FIGURE 25.4 Discontinuities in the curve indicate the possibility of a programming error.

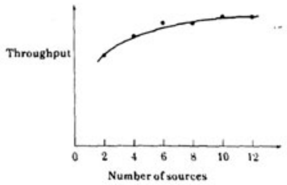


FIGURE 25.5 The curve after verification.

More Verification Techniques

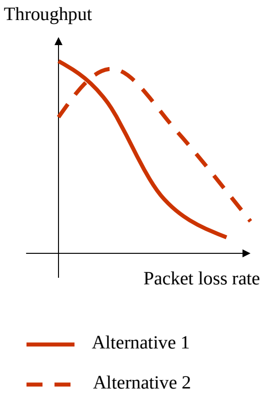
- Degeneracy Tests: Try extreme configuration and workloads
  - One CPU Zero disk
- Consistency Tests: Similar result for inputs that have same effect
  - E.g. Four users at 100 Mbps vs Two at 200 Mbps
  - Build a test library of continuity, degeneracy, and consistency tests
- Seed Independence: Similar results for different seeds

Model Validation Techniques

- **Validation techniques for one problem may not apply to another problem**
- **Aspects to Validate**
  1. **Assumptions**
  2. **Input parameter values and distributions**
  3. **Output values and conclusions**
- **Techniques**
  1. **Expert intuition**
  2. **Real system measurements**
  3. **Theoretical results**
- **➔ 9 validation tests**

Expert Intuition

- **Most practical and common way**
- **Experts = Involved in design, architecture implementation, analysis, marketing or maintenance of the system**
- **Selection = fn of Life cycle stage**
- **Present assumption, input, and output**
- **Better to validate one at a time**
- **See if the experts can distinguish simulation vs measurement**



Theoretical Results

- **Analysis = Simulation**
- **Used to validate analysis also**
- **Both may be invalid**
- **Use theory in conjunction with experts intuition**
  - **E.g. Use theory for a large configuration**
- ***"Fully validated model"* is a myth**
- **Can show that the model is not invalid**

Transient Removal

- **Generally steady state performance is of interest (not the transient one)**
  - **➔ Remove the initial part**
- **No exact definition ➔ Heuristics**
  1. **Long Runs**
  2. **Proper Initialization**
  3. **Truncation**
  4. **Initial Data Deletion**
  5. **Moving Average of Independent Replications**
  6. **Batch Means**

Transient Removal Techniques

- **Long Runs**
  - **Wastes resources**
  - **Difficult to insure that it is long enough**
- **Proper Initialization**
  - **Start in a state close to expected steady state ➔ Reduces the length and effect of transient state**

Truncation

- **Assumes variability is lower during steady state**
- **Plot max-min of $n-l$ observation for $l=1,2, \ldots$**
  - **When l^{th} observation is neither the minimum nor maximum transient state ended**
- **Example: 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 9, 10, 11, 9, 10, 11, 9, …**
  - **At $l=9$ ➔ Range = (9, 11), next observation = 10**
- **Sometimes it gives incorrect results**

Truncation – cont’d

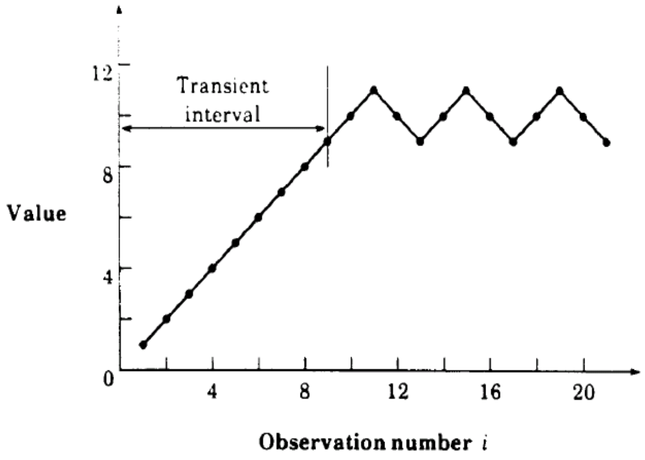


FIGURE 25.7 Plot of the data used in the truncation method example.

Initial Data Deletion

- **Delete some initial observation**
- **Compute average**
- **No change ➔ Steady state**
- **Use several replications to smoothen the**
- **average**
- ***m* replications of size *n* each - x_{ij} j^{th} observation in the i^{th} replication**

Initial Data Deletion – cont'd

- **Procedure**
  1. **Get a mean trajectory by averaging across replications**

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

  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. **Delete the first *l* observations 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, R** $R = \frac{\bar{\bar{x}}_l - \bar{\bar{x}}}{\bar{\bar{x}}}$
  5. **Repeat steps 3 and 4 and by varying *l* from 1 to n**
  6. **Plot the overall mean and the relative change**
  7. ***l* at knee = length of the transient interval**

Initial Data Deletion – cont'd

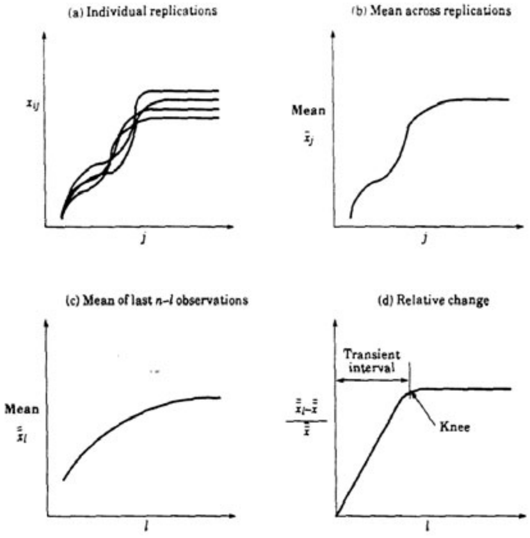


FIGURE 25.8 Initial data deletion method.

Moving Average of Independent Replications

- **Mean over a moving time interval window**
  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, … and so on until the plot is smooth**
  4. **Value of j at the knee gives the length of the transient phase**

Moving Average of Independent Replications – cont’d

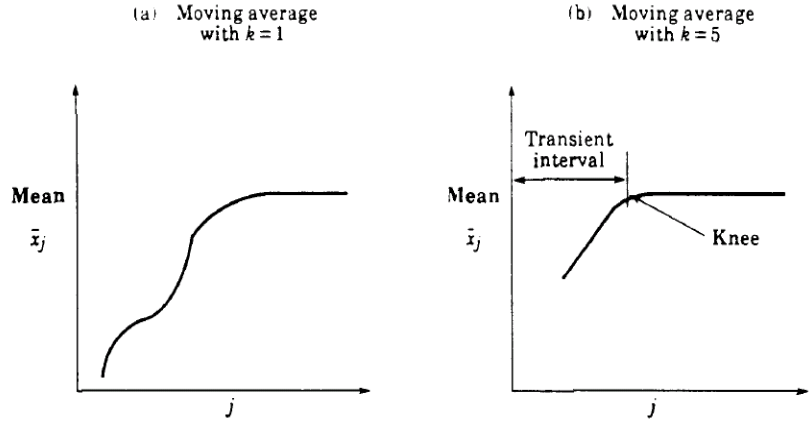


FIGURE 25.9 Moving average of independent replications.

Batch Means

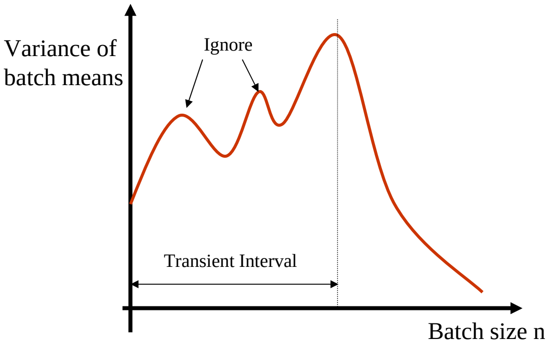
- **Run a long simulation and divide into equal duration part**
- **Part = Batch = Subsample**
- **Study variance of batch means as a function of the batch size**

Batch Means – cont'd

- **Procedure:**
- **For each batch compute a batch mean** $\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \qquad i=1,2,\ldots,m$
- **Compute overall mean** $\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m} \bar{x}_i$
- **Compute the variance of the batch means** $Var(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}\left(\bar{x}_i - \bar{\bar{x}}\right)^2$
- **Repeat steps 1 and 3 for n = 3, 4, 5, …**
- **Plot the variance as a function of batch size n**
- **Value of n at which the variance definitely starts decreasing gives transient interval**

- **Rationale**
  - **Batch size << transient ➔ overall mean = initial mean ➔ Lower variance**
  - **Batch size >> transient ➔ Overall mean = steady state mean ➔ Lower variance**
- **Ignore peaks followed by upswing**

Batch Means – cont'd



Terminating Simulations

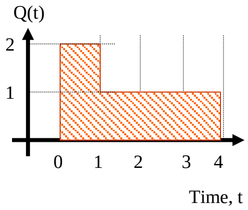
- **Transient performance is of interest**
  - **E.g. Network traffic under specific condition**
  - **System shuts down**
- **➔ Do not need transient removal**
- **Final conditions:**
  - **May need to exclude the final portion from results**
  - **Techniques similar to transient removal**

Treatment of Leftover Entities

- **Mean service time** = $\dfrac{\textbf{Total service time}}{\textbf{Number of jobs that completed service}}$

- **Mean waiting time** = $\dfrac{\textbf{Sum of waiting time}}{\textbf{Number of jobs that received service}}$

- **Mean Queue Length, Qavg** $Q_{avg} = \frac{1}{T}\int_0^T Q(t)dt$



Stopping Criteria: Variance Estimation

- **Run until confidence interval is narrow enough**

$$\bar{x} \pm z_{1-\alpha/2} Var(\bar{x})$$

- **For independent observations** $Var(\bar{x}) = \dfrac{Var(x)}{n}$
- **Independence not applicable to most simulations**
  - **E.g. Large waiting time for i**th **job Large waiting time for i+1**st **job**
- **For correlated observations** $Var(\bar{x}) >> \dfrac{Var(x)}{n}$
- **Solutions (for correlated cases):**
  1. **Independent Replications**
  2. **Batch Means**
  3. **Method of Regeneration**

Independent Replications

- **Assumes that means of independent replications are independent**

- **Procedure:**
  - **(1) Conduct m replications of size n n0 each**
  - **(2) Compute a mean for each replication**

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

  - **(3) Compute an overall mean for all replications**

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

  - **(4) Calculate the variance of replicate means**

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

  - **(5) Confidence interval for the mean response**

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

- **Keep replications to avoid waste – 10 replications typically sufficient**

Batch Means

- **Also called method of subsamples**
- **Procedure:**
  - **(1) Run a long simulation run**
  - **(2) Discard initial transient interval and**
  - **(3) Divide the remaining observations run into several batches or subsamples**
  - **(4) Compute means for each batch**

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

  - **(5) Compute an overall mean**

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

  - **(6) Calculate the variance of batch means**

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

  - **(7) Confidence interval for the mean response is**

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

Batch Means – cont'd

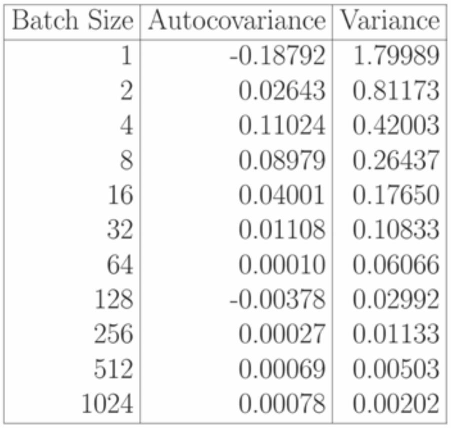
- **Less waste than independent replications**
- **Keep batches long to avoid correlation**
- **Check:**
  - **Compute the autocovariance of successive batch means**

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

  - **Double *n* until autocovariance is small**

Example: Batch Means – cont'd

- **Problem: When to stop?**



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

- **Solution: at n = 64, autocovariance < 1% of sample variance**

Method Of Regeneration

- **Terminology:**
  - **Regeneration point**
  - **Regeneration cycle**
  - **Regenerative system**
  - **Nonregenerative long memory**
    - **E.g. network of queues**

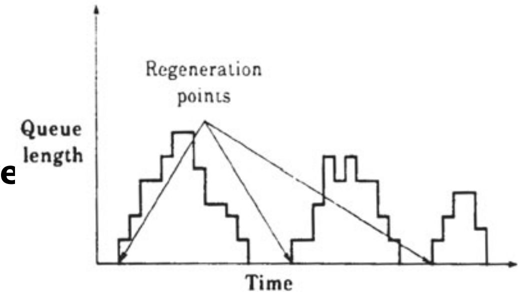


FIGURE 25.13 Regeneration points.

Method Of Regeneration – cont’d

- **Behaviour after idle period does not depend upon the past history**
  - **System takes a new birth ➔ regeneration point**
- **Regeneration cycle: Between two successive regeneration points**
- **Use means of regeneration cycles**
- **Problems:**
  - **Not all systems are regenerative**
  - **Different lengths ➔ Computation complex**
  - **Overall mean ≠ Average of cycle means**

Method Of Regeneration – cont'd

- **Suppose we have a regenerative simulation consisting of m cycles of sizes: n_1, n_2, …, n_m.**
- **Procedure:**
  - **Cycles' means are given by** $\bar{x}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij}$
  - **Overall mean can be calculated as**
    - **Compute cycle sums:**

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

    - **Compute the overall mean**

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

Note: The overall mean is NOT equal to

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

Method Of Regeneration – cont'd

- **Procedure (cont'd):**
  - **Calculate the difference between expected and observed cycle sums:**

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

  - **Calculate the variance of the differences:**

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

  - **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**

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

Method Of Regeneration – cont’d

- **Advantages:**
  - **Does not require removal of transient observations**

- **Disadvantages:**
  - **Cycle lengths are unpredictable**
  - **Finding the regeneration points is NOT trivial**
    - **May require lots of checking after every event in the simulation**
  - **Many of the variance reduction techniques such as common random streams or antithetic variables can not be used due to the variable length of cycles**
  - **Mean and variance estimators are biased in the sense that their expected values from a random sampling are not equal to the quantity being estimated**