Analysis of Simulation Results

25-1

Overview

- Analysis of Simulation Results
- Model Verification Techniques
- Model Validation Techniques
- Transient Removal
- Terminating Simulations
- Stopping Criteria: Variance Estimation

25-2

Model Verification vs. Validation

- Verification ⇒ Debugging
- Validation ⇒ Model = Real world

- Four Possibilities:
  1. Unverified, Invalid
  2. Unverified, Valid
  3. Verified, Invalid
  4. Verified, Valid

25-3

Model Verification Techniques

1. Top Down Modular Design
2. Anti-bugging
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

25-4

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 sub-modules

25-5

Verification Techniques

- **Anti-bugging**: Include self-checks:
  $\sum$ 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

25-6

Trace

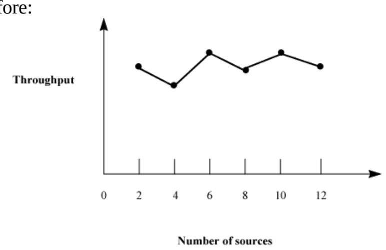
- Trace = Time-ordered list of events and variables
- Several levels of detail:
  - Events trace
  - Procedure trace
  - Variables trace
- User selects the detail
  - Include on and off
- See Fig 25.3 in the Text Book on page 418 for a sample trace

On-Line Graphic Displays

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

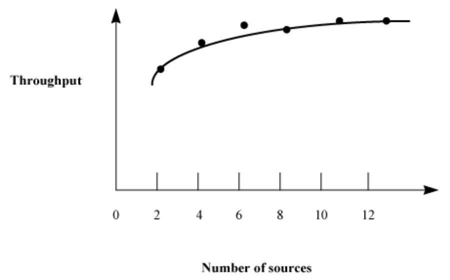
Continuity Test

- Run for different values of input parameters
- Slight change in input ⇒ slight change in output
- Before:



Continuity Test (Cont)

- After:



More Verification Techniques

- **Degeneracy Tests**: Try extreme configuration and workloads
  - One CPU, Zero disk
- **Consistency Tests**:
  - Similar result for inputs that have same effect
    - 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

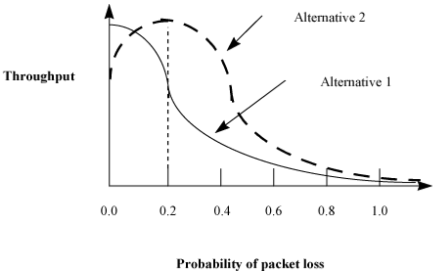
- Validation techniques for one problem may not apply to another problem.
- Aspects to Validate:
  1. Assumptions
  2. Input parameter values and distributions
  3. Output values and conclusions
- Techniques:
  1. Expert intuition
  2. Real system measurements
  3. Theoretical results

⇒ 3 **£** 3 = 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, output
- Better to validate one at a time
- See if the experts can distinguish simulation vs. measurement

Expert Intuition (Cont)



Real System Measurements

- Compare assumptions, input, output with the real world
- Often infeasible or expensive
- Even one or two measurements add to the validity

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
  - Can show that the model is not invalid

Transient Removal

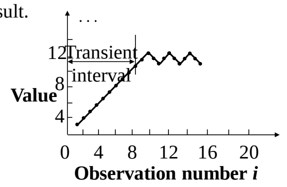
- Generally steady state performance is interesting
- Remove the initial part
- No exact definition ⇒ Heuristics:
  1. Long Runs
  2. Proper Initialization
  3. Truncation
  4. Initial Data Deletion
  5. Moving Average of Independent Replications
  6. Batch Means

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, ……….
- When *(l+1)*th observation is neither the minimum nor maximum ⇒ transient state ended
- At l = 9, Range = (9, 11), next observation = 10
- Sometimes incorrect result.



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}= jth observation in the ith replication

Initial Data Deletion (Cont)

Steps:

1. Get a mean trajectory by averaging across replications

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

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.

Initial Data Deletion (Cont)

3. Delete the first *l* observations and get an overall mean from the remaining *n-l* values:

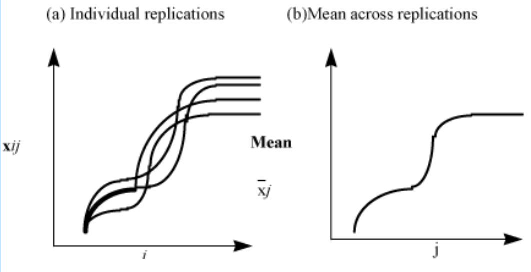
$$\bar{\bar{x}}_l = \frac{1}{n-l}\sum_{j=l+1}^{n} \bar{x}_j$$

4. Compute the relative change:

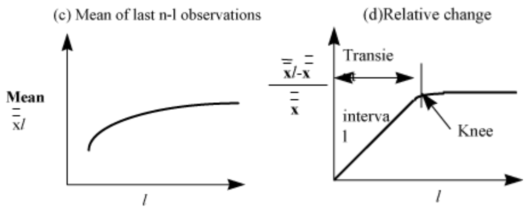
$$\text{Relative change} = \frac{\bar{\bar{x}}_l - \bar{\bar{x}}}{\bar{\bar{x}}}$$

5. Repeat steps 3 and 4 by varying *l* from 1 to *n-1*.
6. Plot the overall mean and the relative change
7. *l* at knee = length of the transient interval.

Initial Data Deletion (Cont)



Initial Data Deletion (Cont)



Moving Average of Independent Replications

- Mean over a moving time interval window

1. Get a mean trajectory by averaging across replications:

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

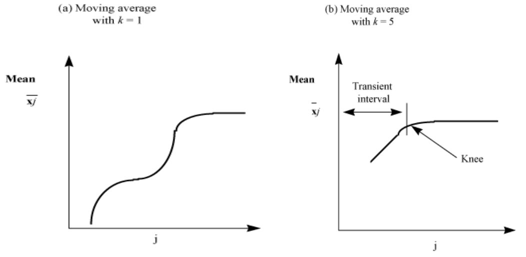
Set k = 1 and proceed to the next step.

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

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

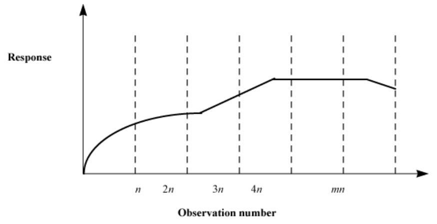
Moving Avg. of Independent Repl. (Cont)

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



Batch Means

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



Batch Means (cont)

Steps:

1. For each batch, compute a batch mean:

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

2. Compute overall mean:

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

3. Compute the variance of the batch means:

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

4. Repeat steps 1 and 3, for n=3, 4, 5, and so on.

Batch Means (Cont)

5. Plot the variance as a function of batch size *n*.
6. Value of *n* at which the variance definitely starts decreasing gives transient interval
7. Rationale:

-Batch size ¿ transient

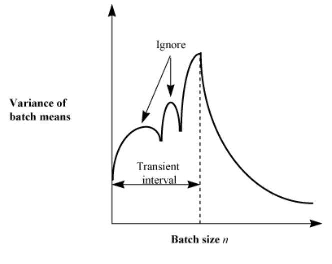
⇒ overall mean = initial mean ⇒ Higher variance

-Batch size À transient

⇒ Overall mean = steady state mean ⇒ Lower variance

Batch Means (Cont)

- Ignore peaks followed by an upswing



Terminating Simulations

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

Summary

1. Verification = Debugging
   ⇒ Software development techniques
2. Validation ⇒ Simulation = Real ⇒ Experts involvement
3. Transient Removal: Initial data deletion, batch means
4. Terminating Simulations = Transients are of interest
5. Stopping Criteria: Independent replications, batch means, method of regeneration