Analysis of Simulation Results

Adapted by Prof. Gheith Abandah

- Model Verification Techniques
- Model Validation Techniques
- Transient Removal
- Terminating Simulations

Model Verification vs. Validation

- Verification $\Rightarrow$ Debugging
- Validation $\Rightarrow$ Model = Real world

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

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

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

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

Trace

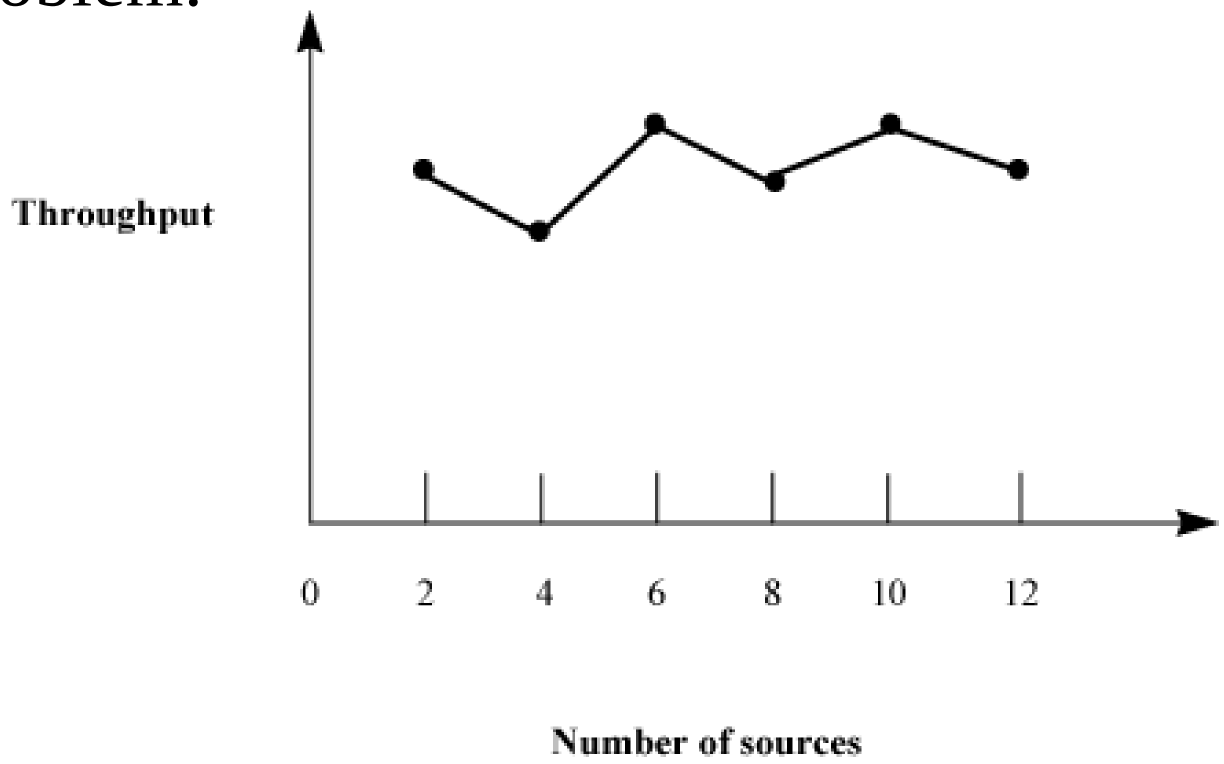
- Trace = Time-ordered list of events and variables
- Several levels of detail:
  - Events trace
  - Procedure trace
  - Variables trace
- User selects the detail
  - Include on and off

On-Line Graphic Displays

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

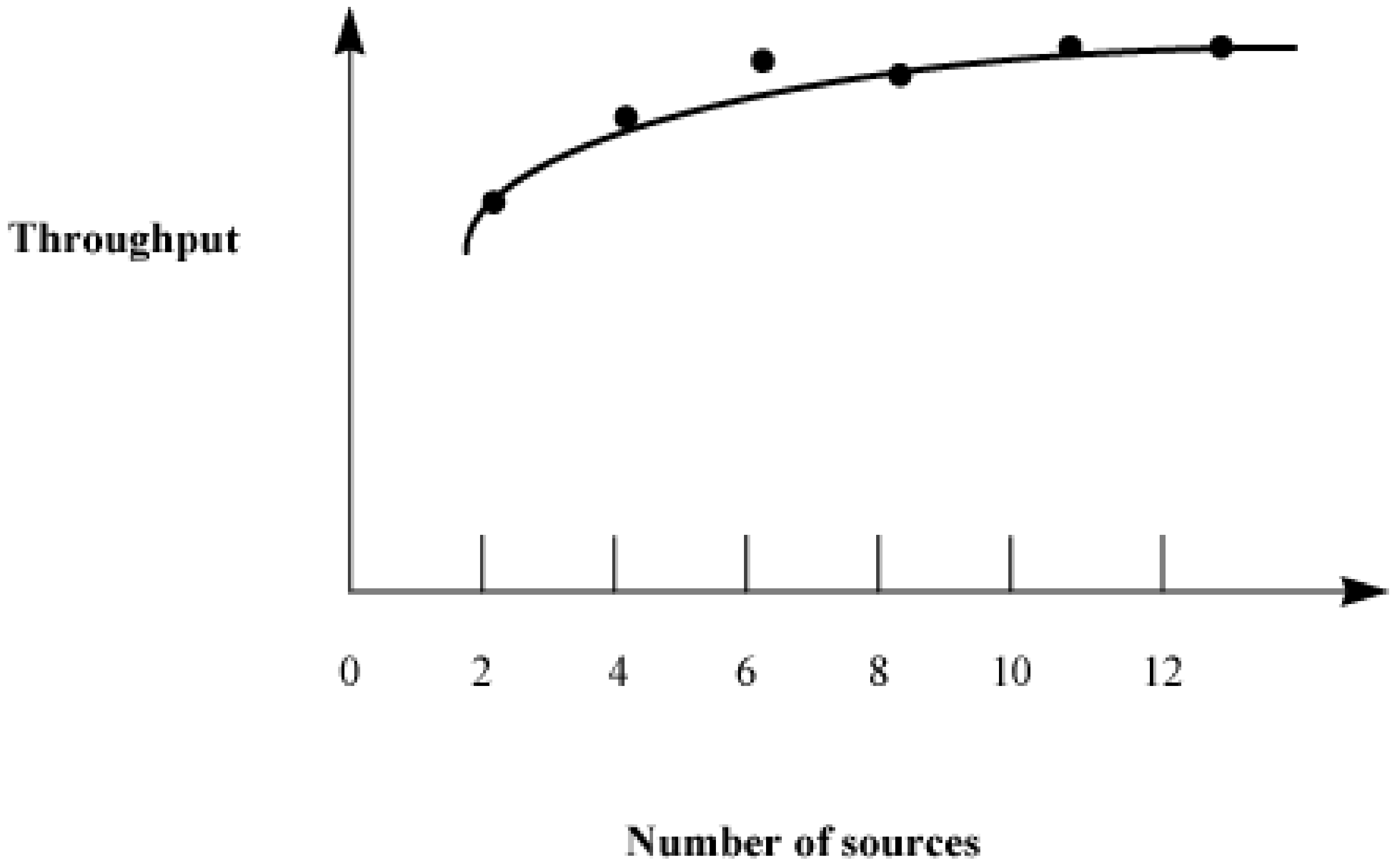
Continuity Test

- Run for different values of input parameters
- Slight change in input $\Rightarrow$ slight change in output
- Example Problem:



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

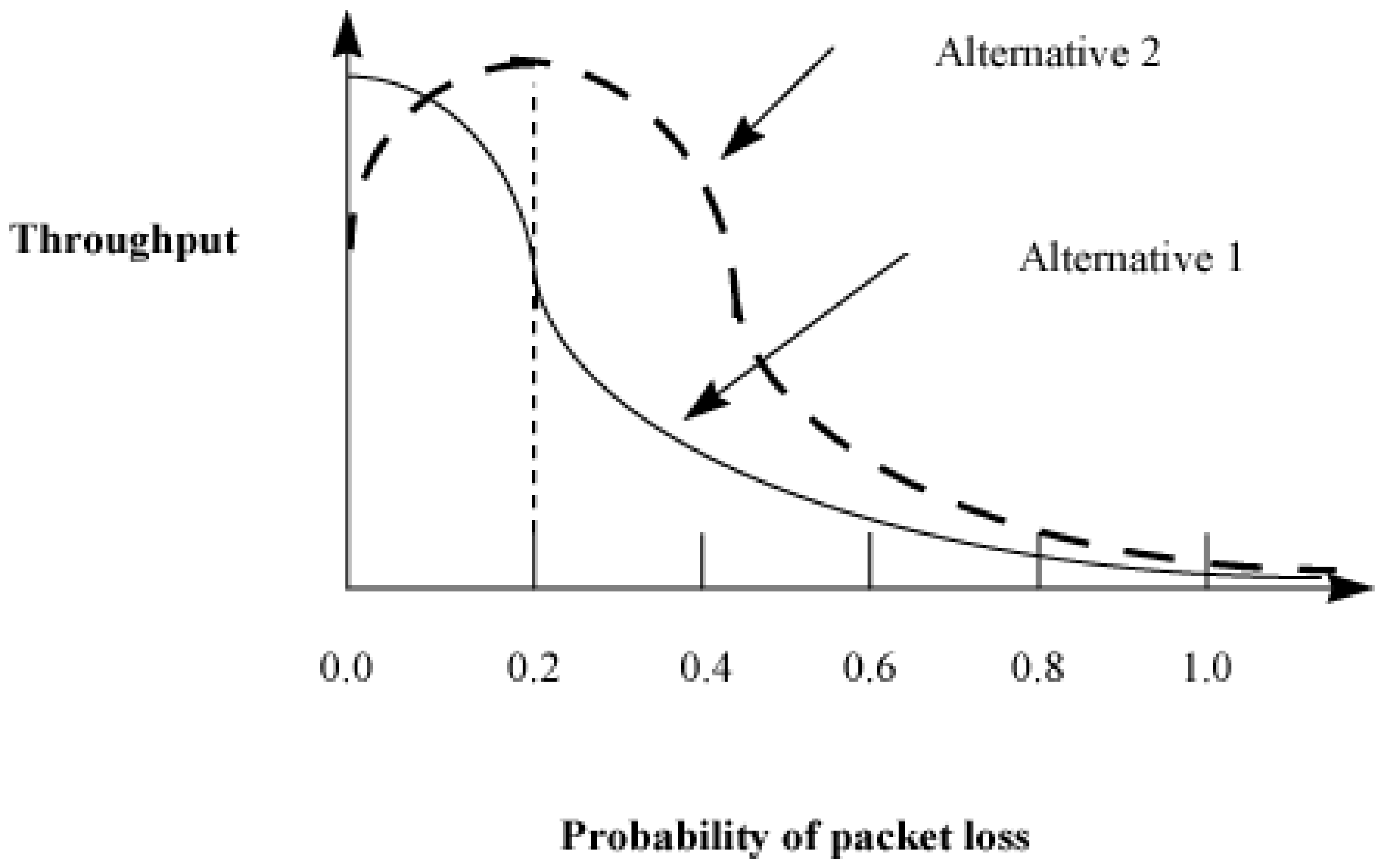
Model Validation Techniques

- 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

Expert Intuition

- Most practical and common way
- Experts = Involved in design, architecture, implementation, analysis, marketing, or maintenance of the system
- Expert Selection = function 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 (Example)



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
- Solutions:
  1. Long Runs
  2. Proper Initialization
  3. Truncation
  4. Initial Data Deletion
  5. Moving Average of Independent Replications

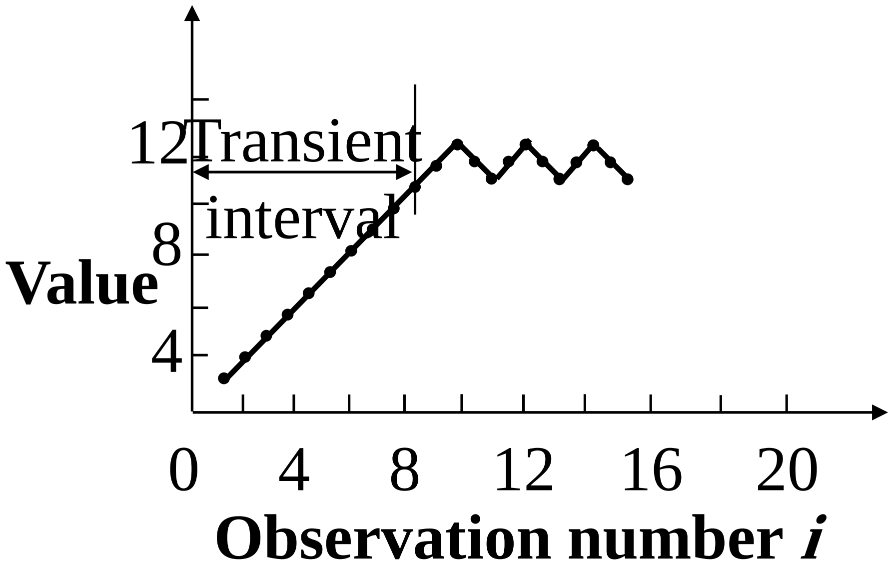
Transient Removal Techniques

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

Transient Removal Techniques

- **Truncation**
  - Assumes variability is lower during steady state
  - Find the transient length *l*

- **Initial Data Deletion**
  - Delete some initial observation



Moving Average of Independent Replications

- Mean over a moving time interval window

1. Get a mean trajectory by averaging across replications:

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

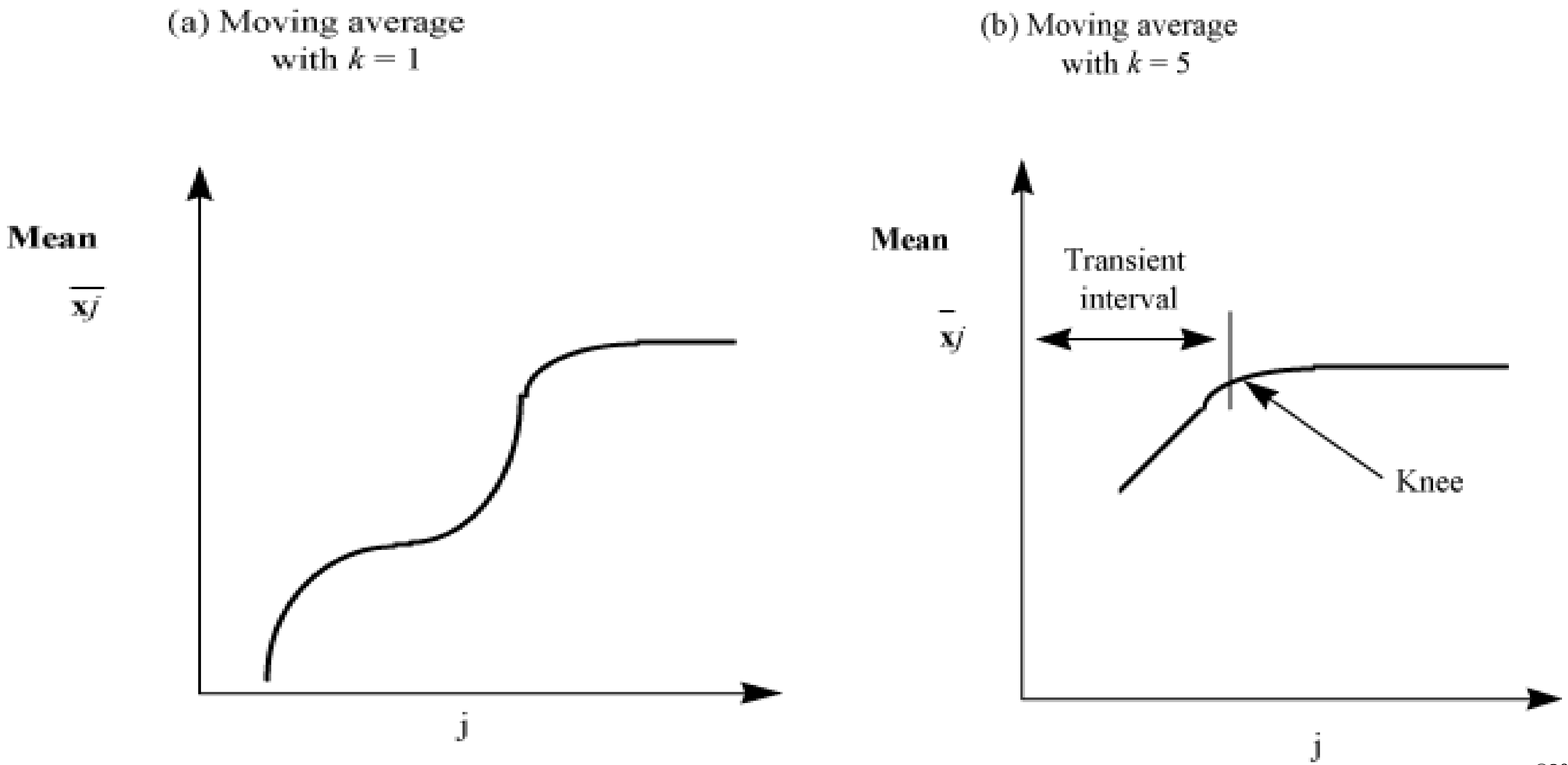
   Set $k = 1$ and proceed to the next step.

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

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

Moving Avg. of Independent Repl. (Cont)

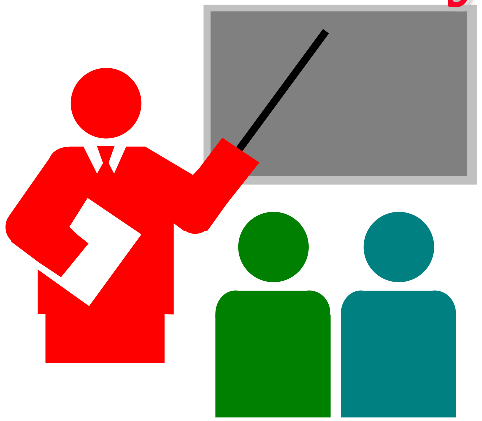
3. Repeat step 2, with k =2, 3, and so on until the plot is smooth.
4. Value of j at the knee gives the length of the transient phase



Terminating Simulations

- Transient performance is of interest
  E.g., Network traffic
- System shuts down $\Rightarrow$ 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
4. Terminating Simulations = Transients are of interest