

SIMULATION

Gaia Maselli
maselli@di.uniroma1.it



SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

- What is a network simulator
 - Software tool to model network protocols (wired and wireless)
- Goal:
 - Build a system which evolves like the real system



When

- Study a complex system (ex. TCP wireless systems)
- Performance evaluation before building a prototype
- Validate analytical study
- Used especially in research
 - Design of new protocols
 - Protocols comparison
- Teaching tool (simulation help in better understanding)



Motivation

- **Only one workstation** is required to run a simulation
- It is possible to analyze a large variety of scenarios
- It is possible to simulate complex topologies (difficult and expensive in testbeds)
 - Nodes mobility
- Easy to test the impact of different versions of a protocol



Pros and Cons

- Pros
 - Evaluate system performance before system development (prototype)
 - Easier to debug
 - Evaluate system scalability
 - Identification of system vulnerabilities
 - Flexibility in studying system behavior
- Cons
 - Detailed comprehension of simulation tool
 - Difficult to study some aspects of the simulated system (e.g., performance of single nodes in network simulator)

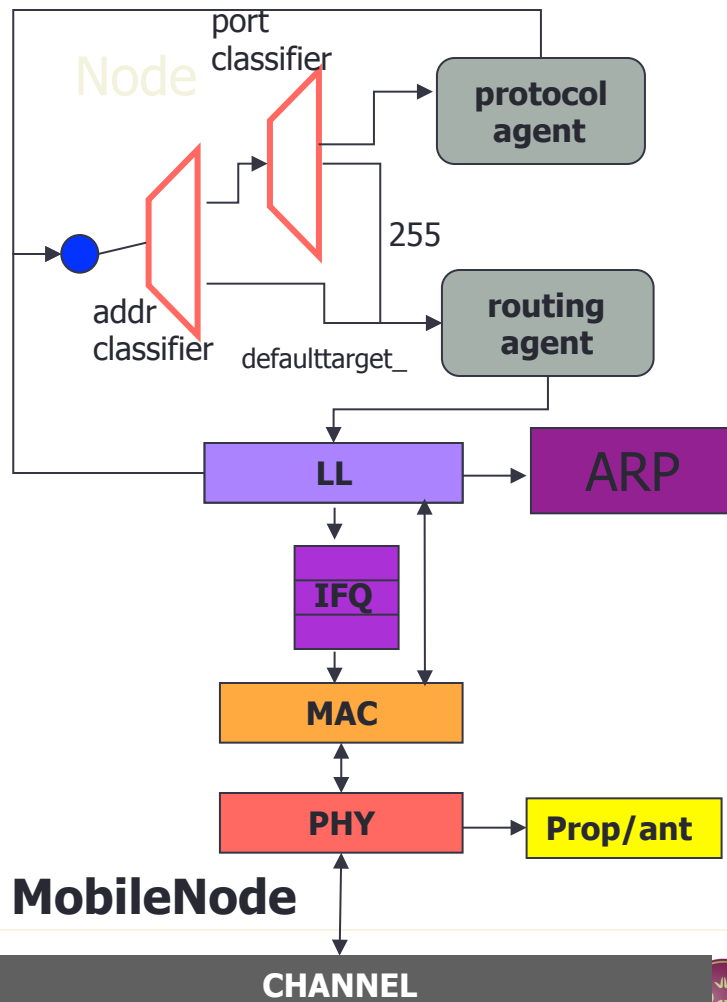


Some simulators

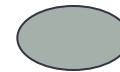
- Network simulator (NS2, NS3)
- OPTNET
- OMNeT++



Wireless node in NS2



Classifier: Forwarding



Agent: Protocol Entity



Node Entry



LL: Link layer object



IFQ: Interface queue



MAC: Mac object



PHY: Net interface



Radio propagation/
antenna models



Analysis of simulation results



Analysis of simulation results

- Model development
 - *Model verification* or debugging (correctly implemented)
 - *Model validation* (represents the real system)
- *Transient removal* (How many of the initial observation should be discarded to ensure that the model has reached a steady state?)
- *Stopping criterion* (How long to run the simulation?)



Model verification techniques

- **Top-down modular design**
 - Simulation models are large computer programs (difficult to develop and debug)
 1. Modularity allows the verification of the simulation to be broken down into smaller problems of verifying the modules and their interfaces
 2. Top-down design consists of developing a hierarchical structure for the model such that the problem is recursively divided into a set of smaller problems.
- **Antibugging:** including additional check and outputs in the program that will point out the bugs, if any
 - Ex, counting the entities generated and served (packets sent and received)



Model verification techniques (Cont)

• Structured walk-Through

- Explaining the code (line by line) to another person or group helps finding bugs


• Run simplified cases

- The model may be run with simple cases: one packet, or only one source, or only one intermediate node

• Event trace

- Trace lists the time, event code, and various packets characteristics associated with the event
- NS outputs traces of the following format:

<event> <time> <_node num_><layer> --- <seq. num> <pkt type> <pkt size> <mac info> --<src dst ttl info><tcp info>



```

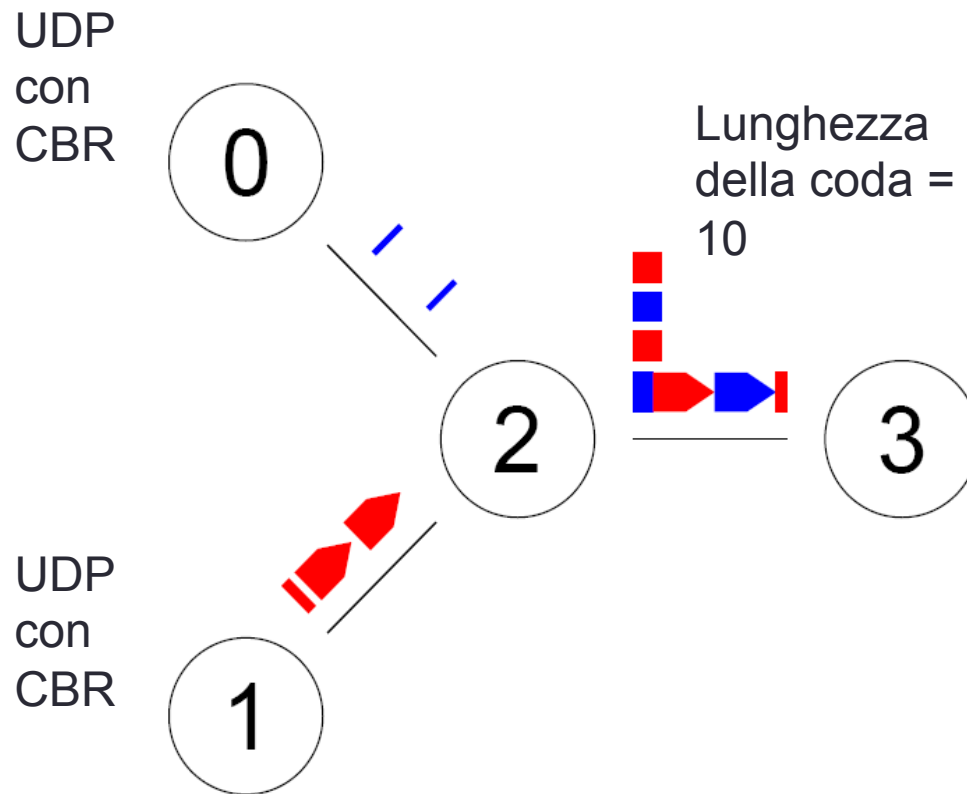
s 60.314477381 _2_ AGT --- 801 tcp 1040 [0 0 0 0] ----- [2:1 9:1 32 0] [1 0] 0 0
s 60.314477381 _2_ AGT --- 802 tcp 1040 [0 0 0 0] ----- [2:1 9:1 32 0] [2 0] 0 0
r 60.344950681 _9_ AGT --- 801 tcp 1060 [13a 9 a 800] ----- [2:1 9:1 254 9] [1 0] 2 0
r 60.355489347 _9_ AGT --- 802 tcp 1060 [13a 9 a 800] ----- [2:1 9:1 254 9] [2 0] 2 0
s 60.355489347 _9_ AGT --- 803 tcp 40 [0 0 0 0] ----- [9:1 2:1 32 0] [2 0] 0 0
  
```



Model verification techniques (Cont)

- **On-Line graphic displays**

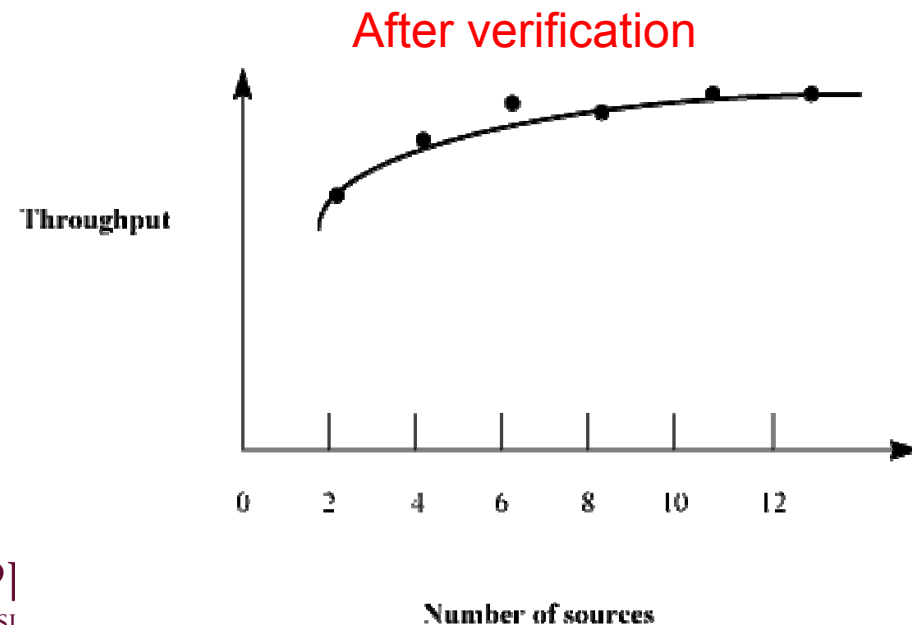
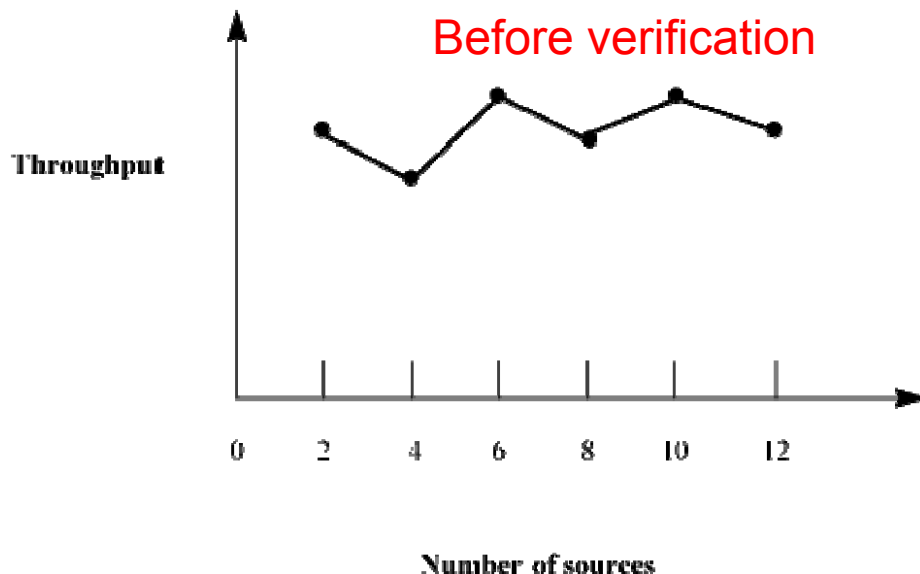
- They present the same information as the trace but in a more comprehensive form



Model verification techniques (Cont)

- **Continuity test**

- Running the simulation several times for slightly different values of input parameters
- For any one parameter a slight change in input should generally produce only a slight change in the output
- Any sudden changes in the output should be investigated (often they are due to modeling errors)



Model verification techniques (Cont)

- **Degeneracy tests**

- Checking that the model works for extreme (lowest or highest allowed) values of system, configuration, or workload parameters

- **Consistency tests**

- Checking that the model produces similar results for input parameter values that have similar effects
- Ex: two sources with an arrival rate of 100 packets per second should load the network to approximately the same level of four sources with an arrival rate of 50 packets per second each
- If the model output shows a significant difference, either the difference should be explained or it could be due to programming errors

- **Seed Independence**

- The seeds used in random-number generation should not affect the final conclusion. The model should produce similar results for different seed values



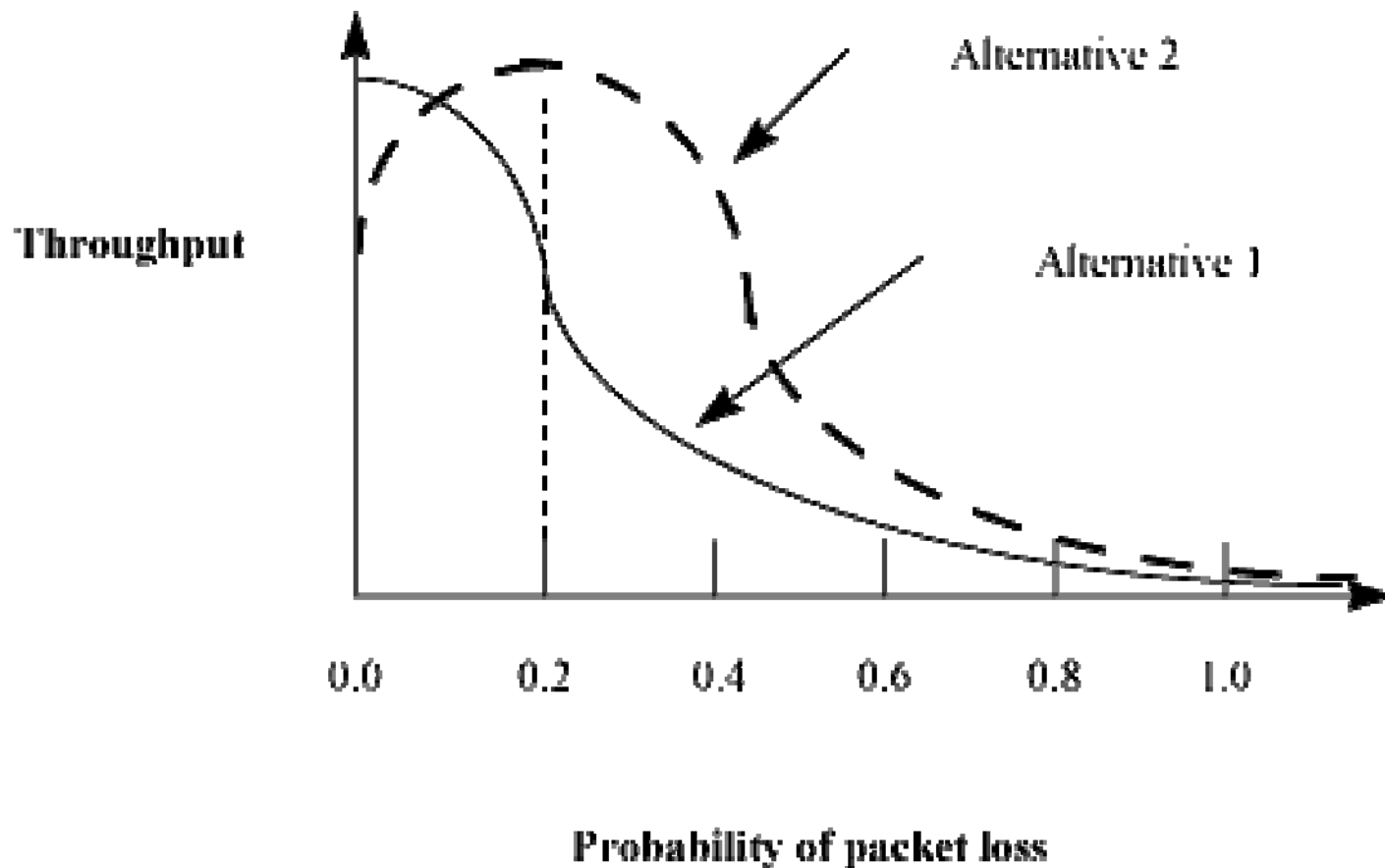
Model validation techniques

- Validation refers to ensuring that the assumptions used in developing the model are reasonable in that, if correctly implemented, the model would produce results close to that observed in real systems
- Model validation consists of validating the three key aspects of the model
 1. Assumptions
 2. Input parameter values and distributions
 3. Output values and conclusions
- How should you validate a model?



Model validation techniques

Expert intuition



Model validation techniques

Real-system measurements

- Comparison with real systems is the most reliable and preferred way to validate a simulation model
- Often unfeasible
 - The real system may not exist
 - Measurements may be too expensive to carry out
- The statistical techniques two test whether two systems are identical may be used to compare the simulation model and the measured data



Model validation techniques

Theoretical results

- In some cases it is possible to
 - analytically model the system under simplifying assumptions
 - Analytically determine what the input distributions should be
- The similarity of theoretical results and simulation results is used to validate the simulation model

In any case, a “fully validated model” is a myth !



Transient removal

How should you determine that the simulation has reached a steady state?



Transient removal

- In most simulations, only the **steady-state performance** (the performance after the system has reached a stable state) is of interest
- Results on the initial part (**transient state**) of the simulations should not be included in the final computations
- **Transient removal**: the problem of identifying the end of the transient state
- It is NOT possible to define exactly what constitutes the transient state and when the transient state ends
- We have to proceed heuristically!



Transient removal methods

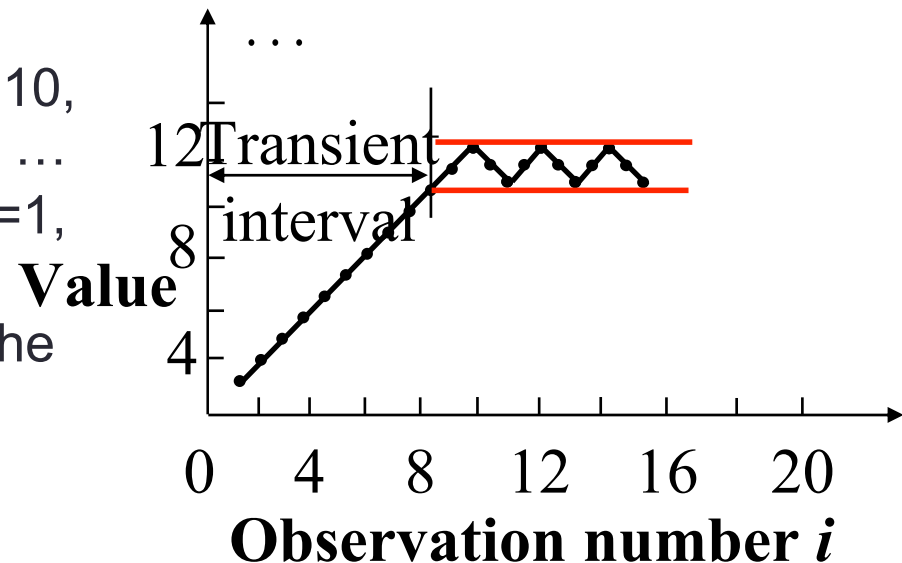
- To use very **long runs**: long enough to ensure that the presence of initial conditions will not affect the results
 - Disadvantages: waste of resources and difficult to understand when long enough
- **Proper initialization**
- Start in a state close to expected steady state (ex. some packets in the queue)
- \Rightarrow Reduces the length and effect of transient state



Transient removal methods

Truncation

- Assumption: the variability during steady state is less than during the transient state
- Variability is measured in terms of range – the minimum and maximum of observations
- Example
- 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, ...
- Plot max-min of $n-l$ observation for $l = 1, 2, \dots$
- When $(l+1)th$ observation is neither the minimum nor maximum \Rightarrow transient state ended
- At $l = 9$, Range = (9, 11), next observation = 10



Transient removal methods

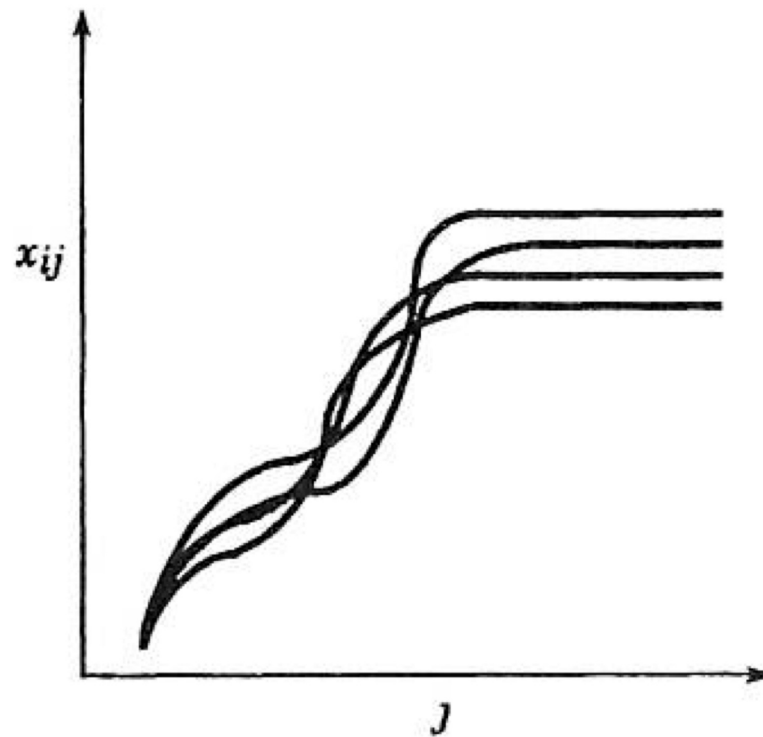
Initial data deletion

- Requires studying the overall average after some of the initial observations are deleted from the sample
- Delete some initial observation
- Compute average
- No change \Rightarrow Steady state
- The randomness in the observations does cause the averages to change slightly even during the steady state
- Use several *replications* to smoothen the average (no change in the input parameters, only in the seed value used in the random-number generator)
- m replications of size n each
- x_{ij} = j -th observation in the i -th replication



Transient removal: Initial data deletion

(a) Individual replications



Transient removal: Initial data deletion

Steps:

1. Get a mean trajectory by averaging across replications

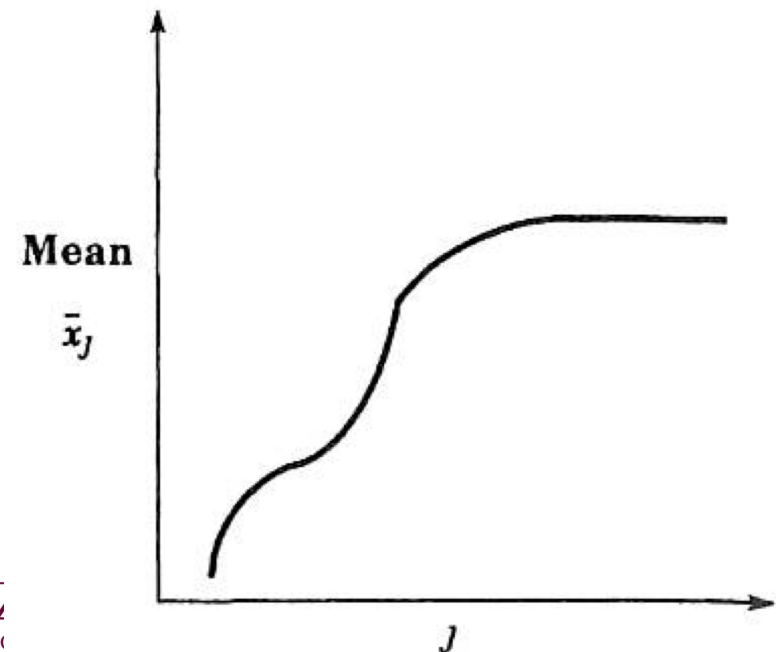
$$\bar{x}_j = \frac{1}{m} \sum_{i=1}^m x_{ij} \quad j = 1, 2, \dots, n \quad (\text{observations})$$

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.

(b) Mean across replications



Transient removal: Initial data deletion

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:

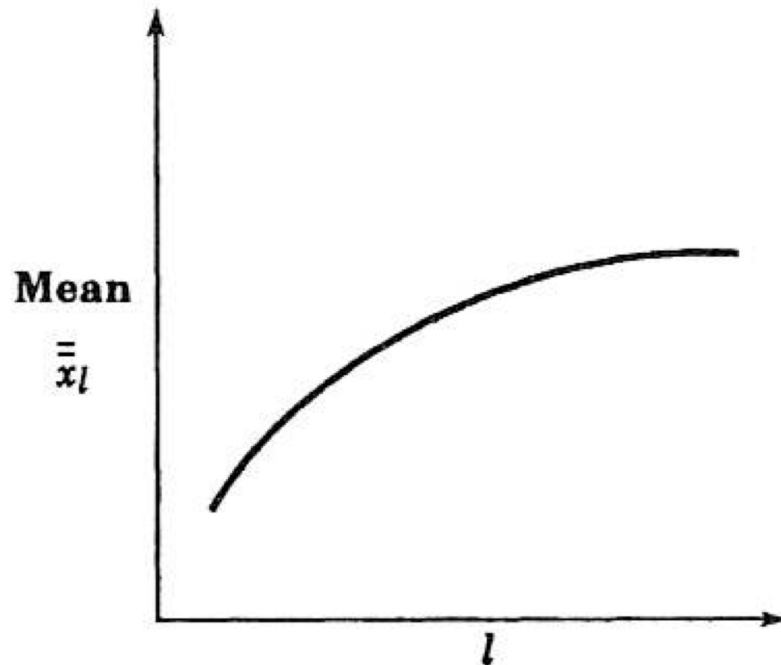
$$\text{Relative change} = \frac{\bar{\bar{x}}_l - \bar{\bar{x}}}{\bar{\bar{x}}}$$

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

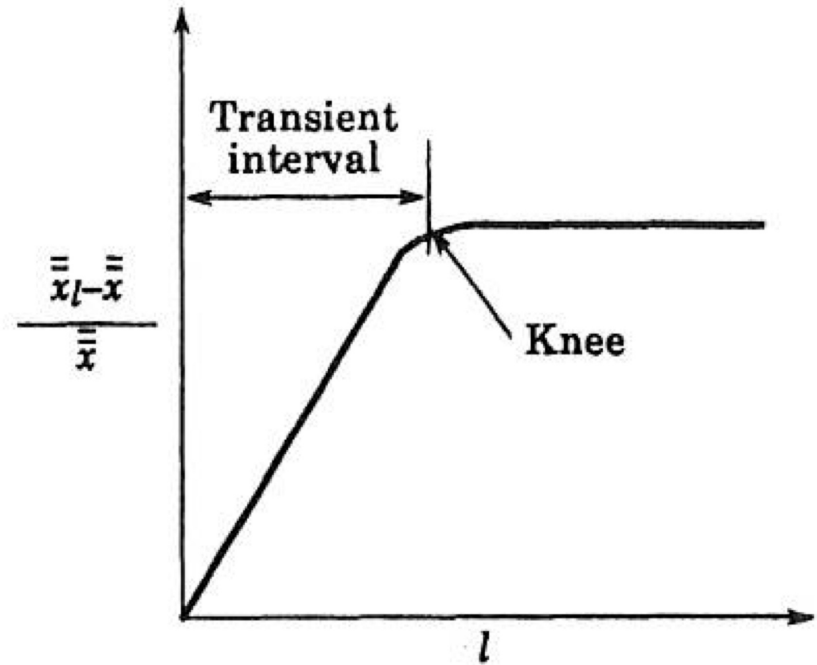


Transient removal: Initial data deletion

(c) Mean of last $n-l$ observations



(d) Relative change



Terminating simulations

- There are systems that never reach a steady state
 - These systems always operate under transient conditions
 - Some systems shut down
 - Some systems have parameters that change with time
- Do not require transient removal



Final conditions

- Conditions at the end of the simulation
- The system state at the end of the simulation may not be typical of the steady state
- It is necessary to exclude the final portion of the response from the steady-state computation
- The methods are similar to those used for determining initial transient periods
- Example: packets in queues at the end of the simulation (they are sent but not received)



Stopping criteria

How long should you run a simulation?



Stopping criteria: Variance estimation

- Too short simulation → the results may be highly variable
- Too long simulation → waste of computing resources and manpower
- The simulation should run until the confidence interval for the mean response narrows to a desired width
- However in simulations are often not independent
- Thus we have to run several replications



Independent replications

- Replications are obtained by repeating the simulation with a different *seed* value (random number generation)
- The means of independent replications are independent even though observations in a single replication are correlated
- The method consists of conducting m replications of size $n+n_0$ each, where n_0 is the length of the transient phase
- The first n_0 observations of each replication are discarded
- The remaining steps are:



Independent replications

1. Compute a mean for each replication:

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

2. Compute an overall mean for all replications:

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



Independent replications

3. Calculate the variance of replicate means:

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

4. Confidence interval for the mean response is:

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

- ❑ Keep replications large to avoid waste
- ❑ Ten replications generally sufficient

