Introduction to Simulation

Intensive Computation - 2015/16

Annalisa Massini

Slides prepared by using:

- Slides from **Simulation modeling and analysis,** A.M. Law & W.D. Kelton - Chapter 1 (http://cs.wpunj.edu/~kaufmanl/cs404/cs404.html

  or

  http://www.statru.org/wp-content/uploads/2010/12/9902_MAK_Basic_Sim.pdf)

- Lecture notes of John Mellor-Crummey *Introduction to Simulation* and *Analyzing Simulation Results* (http://www.cs.rice.edu/~johnmc/comp528/lecture-notes/index.html)

Simulation

- **Simulation**: Imitate the operations of a facility or process, usually via computer
  - What's being simulated is the ***system***
  - To study system, often make assumptions/approximations, both logical and mathematical, about how it works
  - These assumptions form a ***model*** of the system
  - If model structure is simple enough, could use mathematical methods to get exact information on questions of interest — *analytical solution*

Simulation

- **Models** of large systems are usually very complex
  - now have more general, flexible modeling sw
- **Simulation** can consume a lot of computer time
  - now have faster, bigger, cheaper hardware to allow for much better studies
  - simulation will continue to need more computing power for more accurate simulation models
- **Simulation** is not "just programming"
  - There's a lot more to a simulation study than just "coding" a model in some software and running it
  - Need careful design and analysis of simulation models – simulation methodology

Systems, Models, and Simulation

- **System**: A collection of entities (people, particles, components, messages, servers, …) that act and interact together toward some end
  - Depends on **objectives** of study
  - **Boundaries** (physical and logical) of the system
  - Level of **detail** (e.g., what is an entity?)
  - Usually assume a **time** element – *dynamic* system
- **State** of a system: collection of variables and their values necessary to describe the system at that time
  - Might depend on desired objectives, output performance measures

Systems, Models, and Simulation

Types of systems

- **Discrete**
  - State variables change instantaneously at separated points in time
  - Example: Bank model → State changes occur only when a customer arrives or departs
- **Continuous**
  - State variables change continuously as a function of time
  - Example: Airplane flight → State variables (like position, velocity) change continuously

Many systems are partly discrete, partly continuous

Systems, Models, and Simulation

Classification of simulation models

- *Static* vs. *dynamic*
- *Deterministic* vs. *stochastic*
- *Continuous* vs. *discrete*

Most operational models are

dynamic, stochastic, and discrete

will be called ***discrete-event simulation models***

Topics

- **The role of simulation**
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- Types of simulations
- Model verification and model validation
- Transient removal
- Terminating simulations
- Time-advance Mechanisms

The Role of Simulation

Why simulation?

- system under study may not be available
  - common in design and procurement stages
- simulation may be preferred alternative to measurement
  - controlled study of wider range of workloads and environments
- higher accuracy results than analytical modeling

Why not?

- accurate simulation models take a long time to develop
  - typically the evaluation strategy that takes the longest

Evaluation criterium

- Combining evaluation techniques is useful
  - analytical model: find interesting range of parameters
  - simulation: study performance within parameter range
- Until validated, all evaluation results are suspect!
  - always validate one analysis modality with another
  - beware of counterintuitive results!

Topics

- The role of simulation
- **Common mistakes in simulation**
- Causes of simulation failure
- Simulation terminology
- Types of simulations
- Model verification and model validation
- Transient removal
- Terminating simulations
- Time-advance Mechanisms

Too Much Detail

- **Level of detail** limited only by time available for development
- A **detailed model** may not be a better model
  - may require **more detailed knowledge** of input parameters
    - inaccurate assumptions can yield wrong results
    - example: time to service disk requests for timesharing simulation
      - could use exponential distribution for time for request service
      - could simulate disk rotation and head movement
      - but simulation better only if sector & track locations known
  - may take **too much time** to develop

Too Much Detail

- Recipe for success
  - start with less-detailed model
  - get some results
  - study **sensitivity**
  - introduce details in key areas that affect results most

Programming Language

- **Programming language** = impact on development time
- **Special-purpose languages**
  - example: Facile [Larus Hill, Schnarr PLDI 2001] - language and compiler for processor simulators
  - require less model development
  - simplify several common tasks, e.g.
    - verification using traces
    - statistical analysis
    - “fast-forwarding” of simulations
- **General-purpose languages**
  - more portable
  - provide more control over efficiency and run-time
  - lack support for model development

Unverified Models

- Simulations are computer programs
- **Bugs** and **programming errors** are common
- Need to **verify models** to avoid wrong conclusions
  - check that the model does what it is intended to do
    - check whether simulation implements assumptions properly

Invalid Models

- Even if simulation has no errors it may not be **representative**
  - assumptions: must validate representativeness
    - otherwise, simulated behavior will not be representative
- All simulation results are suspect
- Must **confirm** with at least one of
  - analytical model
  - measurements
  - intuition

Initial Conditions

Improper handling of initial conditions

- Initial part of a simulation is generally not representative
  - transient behavior rather than steady state
- Initial part of simulation should be discarded
  - several techniques for identifying beginning of steady state

Too Short Simulations

- Simulation run times are often very long
- Temptation is to halt simulations ASAP
- However
  - results may be heavily dependent on initial conditions
  - may not be representative of a real system until steady state
- Correct length for simulations depends on
  - accuracy desired (width of confidence intervals)
  - variance of observed quantities

Bad Random Numbers

- Bad random numbers can pollute simulation results
- How can random numbers be bad?
  - period too short
  - assume global randomness = local randomness
  - rely on bit subsets: may not be as random as whole
- Rule of thumb
  - use well-known generator rather than rolling your own
- Even well-known generators have had problems

Bad Random Numbers

- Improper selection of RNG seeds
  - seeds for different random streams must be carefully chosen
    - must ensure independence of streams
    - sources of error
      - share one stream for several different processes
      - use same seed for all streams
- Impact: introduce correlation among processes that may lead to non-representative results

Topics

- The role of simulation
- Common mistakes in simulation
- **Causes of simulation failure**
- Simulation terminology
- Types of simulations
- Model verification and model validation
- Transient removal
- Terminating simulations
- Time-advance Mechanisms

Causes of Simulation Failure

- Inadequate time estimate: underestimate effort required
  - often start off as 1-week or 1-month projects
  - continue for years
    - good: more features, parameters to provide better detail
    - bad: add more detail in hope of making it useful
- No achievable goal
  - should have SMART goals
    - specific, measurable, achievable, repeatable, thorough
  - not measurable: to model X
  - projects without goals continue until funding runs out

Causes of Simulation Failure

- Incomplete mix of essential skills for a simulation project
  - project leadership: lead, motivate, manage
  - modeling and statistics: identify and model key characteristics
- at required level of detail
  - programming: construct readable and verifiable program
  - knowledge of modeled system: understand model, interpret results and their implications

Causes of Simulation Failure

- Inadequate level of user participation
  - modeling team and users must discuss system changes
    - most systems change
    - models developed in a vacuum rarely succeed
- Obsolete or nonexistent documentation
  - most simulation models evolve over time as system does
  - if system documentation is obsolete, modeling errors are likely
  - best to use literate programming to keep documentation in sync

Causes of Simulation Failure

- Inability to manage development of large, complex programs
  - tools can help track
    - design objectives
    - functional requirements
    - data structures
    - progress estimates
  - other useful principles
    - top-down design
    - structured programming
  - without tools and techniques, hard to develop large models successfully

Causes of Simulation Failure

Mysterious results

- Causes
  - bugs in simulation program
  - invalid modeling assumptions
  - lack of understanding of system to be modeled
- What to do?
  - attempt to verify the model
  - bring persistent mysterious results to attention of users
    - may lead to unexpected insight into system
    - may point to system features that must be modeled in more detail

Analysis of Simulation Failure

Simulation Checklist: Before Development

- Is the **goal** of the simulation properly specified?
- Is the **level of detail** in the model appropriate for the goal?
- Does the team include appropriate personnel?
  - leadership, statistics and modeling, programming, and system
- Has **sufficient time** been allotted for the project?

Analysis of Simulation Failure

Simulation Checklist: During Development

- Has the **random number generator** been tested?
  - uniformity
  - independence
- Is the model **reviewed** regularly with the end user?
- Is the model **documented**?

Analysis of Simulation Failure

Simulation Checklist: During Execution

- Is the simulation **length** appropriate?
- Are 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 random streams will not overlap?

Topics

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- **Simulation terminology**
- Types of simulations
- Model verification and model validation
- Transient removal
- Terminating simulations
- Time-advance Mechanisms

Simulation Terminology

- **State variables**: define the state of system
- **Event**: change in system state
- **Continuous-time vs. discrete-time models**
  - continuous-time model: system state is defined at all times
  - discrete-time model: state defined only at particular instants
- **Continuous-state vs. discrete-state models**
  - classified by type of variables: continuous or discrete
    - continuous: uncountably infinite values
    - discrete: countable
  - AKA continuous-event and discrete event models
- **Deterministic vs. probabilistic models**
  - deterministic: output can be predicted with certainty
  - probabilistic: a different result for same input parameters

Simulation Terminology

- **Static vs. dynamic models**
  - static: time is not a model variable
- **Linear vs. non-linear models**
- **Open vs. closed models**
  - open: input from outside the model
    - queuing model with arcs from outside
  - closed: no external input
- **Stable vs. unstable models**
  - stable: behavior settles down to steady state that is independent of time
  - unstable: continuously changing behavior

Topics

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- **Types of simulations**
- Model verification and model validation
- Transient removal
- Terminating simulations
- Time-advance Mechanisms

Simulation Types

- **Monte Carlo**
- **Trace-driven simulation**
- **Program or Execution-driven simulation**
- **Discret-event simulation**

Monte-Carlo Simulations

- Model probabilistic phenomenon that do not change over time

- Applications
  - simulation of random or stochastic processes
    - complex physical phenomena such as radiation transport
    - sub-nuclear processes in high energy physics experiments traffic flow
  - evaluation of integrals

Monte-Carlo Simulations

- Requirements
  - system can be described by probability density functions
  - good pseudo-random number generator available
- How they are commonly performed
  - given PDFs, simulations proceed by random sampling
  - multiple simulations (trials) are performed
  - desired result is taken as avg over # of observations
  - predictions of variance in avg result used to estimate #trials needed to achieve a given error bound

Trace-driven Simulation

- Trace = time ordered record of events on real system
- Applications: analyze paging, scheduling, caches, etc.
- Advantages
  - credibility: easy to sell
  - easy validation: compare with measured system
  - accurate workload: trace preserves correlation & interference effects
  - detailed tradeoffs: possible to evaluate small changes in model
  - less randomness: deterministic input reduces output randomness
  - fair comparison of alternatives
  - similarity of implementation: model is similar to system

Trace-driven Simulation

- Disadvantages
  - complexity: requires detailed simulation of system
  - representativeness: traces from one system may not be representative
  - finiteness: may not represent much time because of size constraints
  - single point of validation: algorithm good for one trace, not others
  - high level of detail: simulations can be costly
  - hard to evaluate changes in workload characteristics: need another trace
    - no feedback from simulation of changes that effect event ordering

Program and Execution-driven Simulation

- Similar to trace-driven simulation except
  - program under study and simulation are interleaved
    - produce and consume event stream in interleaved fashion

- Key advantages over trace-driven simulation
  - avoids specialized hardware for collecting traces
  - avoids storage of long traces
  - simpler to study new workloads

Discrete-Event Simulations

- Discrete-event simulations use discrete-state model of system
  - e.g. model number of threads queued for various resources
  - may use discrete or continuous time values
- Components
  - event scheduler: linked list of pending events
    - operations: schedule event X at time T; hold event X for time interval dt; cancel previously scheduled event X; hold X indefinitely; schedule indefinitely held event
  - simulation clock: maintains global time
    - unit time or event-driven advancement

Discrete-Event Simulations

- Components
  - system state variables
  - event routines: one for each event type
  - input routines: read model parameters
  - initialization routines for system variables & RNG
  - trace routines: print intermediate results periodically
  - dynamic memory management, usually GC managed storage
  - report generator to calculate and print final result
  - main program: invokes all components in proper order

Discrete-Event Simulations

Event Sets for Discrete-Event Simulations

- Ordered set of future event notices
  - typically an ordered linked list
- Operations
  - insert event
  - find next scheduled event
  - remove next scheduled event
- Choice of data structure affects execution time
  - depends on frequency of insertion/deletion and avg # events
- Possible implementations
  - divide future into indexed intervals of Δt
  - each interval has own sublist
  - tree structures, e.g. heap

Topics

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- Types of simulations
- **Model verification and model validation**
- Transient removal
- Terminating simulations
- Time-advance mechanisms

Model Goodness

- Measuring goodness
  - validation: are assumptions reasonable?
  - verification: does model implement assumptions correctly?
- Possible model states

invalid, unverified	invalid, verified
valid, unverified	valid, verified

- correctly implements bad assumptions
- incorrectly implements good assumptions
- correctly implements good assumptions

Model Verification Techniques

- Strategies for avoiding bugs
  - software engineering
    - top-down design
      - layered (hierarchical) system structure
    - modularity
      - well-defined interfaces
      - unit testing
  - assertions to check invariants
    - e.g., # packets received = # packets sent - # packets lost - # in flight
    - entity accounting
  - structured walk through
- Deterministic models
  - run simulation with known distributions for random variates

Model Verification Techniques

- Simplified test cases with easily analyzed results
- Tracing: events, procedures, variables
- On-line graphical visualizations
  - convey progress of simulation
- Continuity test
  - test simulation with slightly different parameters
  - investigate sudden changes in output

Model Verification Techniques

- Degeneracy tests
  - check model works for extreme cases
  - e.g. networking: no routers, no router delays, no sources, …
- Consistency tests
  - similar results for parameters that should have similar effects
  - e.g. router simulation: 2 sources, rate r ~ 1 source, rate 2r
- Seed independence
  - similar results for different seed values

Model Validation Techniques

- What to check
  - assumptions
  - input parameter values and distributions
  - output values and conclusions
- How
  - expert intuition: most common and practical
  - measurements of real system
    - are simulation results and measurements distinguishable?
      - can use statistical tests, e.g. paired observations
    - verify input distributions, e.g. chi-square test

Model Validation Techniques

- How (continued)
  - theoretical results
    - simplifying assumptions helps
    - validate a few simple cases of theoretical model with simulation or intuition
    - use analytical model to predict complex cases

Caution: myth of a fully-validated model

- generally possible only to prove model not wrong for some cases
- more comparisons increase confidence, but prove nothing!

Topics

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- Types of simulations
- Model verification and model validation
- **Transient removal**
- Terminating simulations
- Time-advance mechanisms

Transient Removal

- **Transient state**: simulation phase before steady state
- **Steady state** performance is usually that of interest
  - e.g. cache performance after cache is "warm"
- **Goal**: exclude transient state before steady state
- **Problem**: identifying end of transient state
- Heuristic approaches for removing transient state
  - long runs
  - proper initialization
  - truncation
  - initial data deletion
  - moving average of independent replications
  - batch means

Transient Removal

- **Long run** = steady state results long enough to dominate effects of initial transients
- **Disadvantages**
  - wastes resources (computer time and real time)
  - difficult to ensure length of run is “long enough”
- *Recommendation*: **avoid this method**

- **Proper initialization** = starting simulation in state close to expected steady state
  - e.g. start CPU scheduling simulation with non-empty job queue
  - e.g. start WWW cache trace-driven simulation with most frequently referenced files in cache
- **Effect**: reduces length of transient behavior

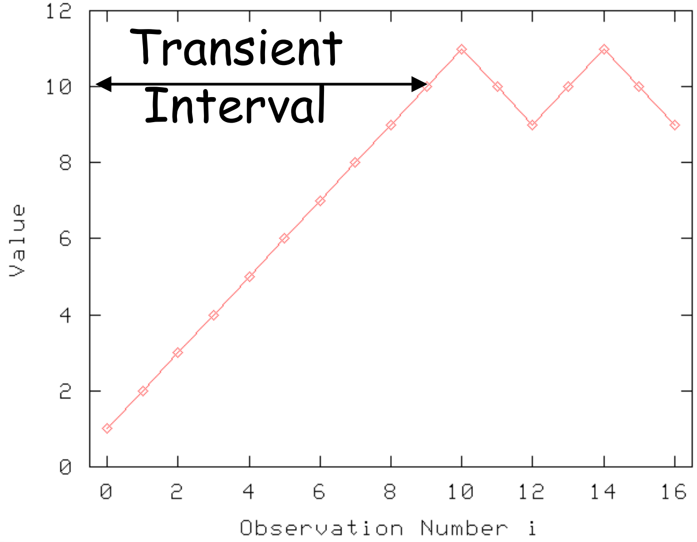
Transient Removal

- **Assumption**: variability of steady state < transient state
- **Truncation method** assumes variability = range
- **Truncation algorithm**

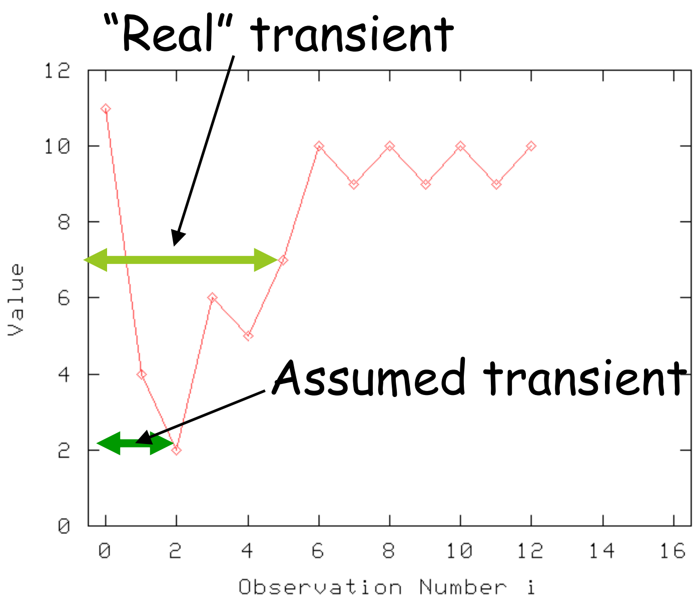
```
input: n observations {x1, x2, …, xn}
for k = 2, n
        mink = min ({xk, …, xn})
        maxk = max ({xk, …, xn})
        if mink ≠ xk && maxk ≠ xk break
post condition: if k ≠ n then k - 1 = length of transient state
```

Transient Removal

Example 1



Example 2



Topics

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- Types of simulations
- Scheduling events
- Model verification and model validation
- Transient removal
- **Terminating simulations**
- Time-advance mechanisms

Terminating Simulations

- Most simulations reach a steady state, but some don't
- Necessary to study such systems in transient state
- Terminating simulations that don't reach steady state
- Other terminating simulations
  - systems with parameters that change over time
  - one that shuts down at 10PM every day
- Terminating simulations don't require transient removal
- Final conditions
  - may not be typical; can apply transition removal conditions to end of simulation

Stopping Criteria

Variance Estimation

- Choosing proper simulation length is important
  - too short: results highly variable
  - too long: wastes time and resources
- Simulation should be run until confidence interval for mean response narrows to desired width

Topics

- The role of simulation
- Common mistakes in simulation
- Causes of simulation failure
- Simulation terminology
- Types of simulations
- Scheduling events
- Model verification and model validation
- Transient removal
- Terminating simulations
- **Time-advance mechanisms**

Time-Advance Mechanisms

- **Simulation clock**: Variable that keeps the current value of (simulated) time in the model
  - Must decide on, be consistent about, time units
  - Usually **no relation** between simulated time and (real) time needed to run a model on a computer

- Two approaches for **time advance**
  - ***Next-event time advance*** (usually used)
  - ***Fixed-increment time advance*** (seldom used)
    - Generally introduces some amount of modeling error in terms of when events *should* occur vs. *do* occur
    - Forces a tradeoff between model accuracy and computational efficiency

Time-Advance Mechanisms

- **Next-event time advance**
  - Initialize simulation clock to 0
  - Determine times of occurrence of future events – *event list*
  - Clock advances to next (most imminent) event, which is executed
    - Event execution may involve updating event list
  - Continue until stopping rule is satisfied (must be explicitly stated)
  - Clock "jumps" from one event time to the next, and doesn't "exist" for times between successive events … periods of inactivity are ignored