1.1 THE NATURE OF 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*

1.1 The Nature of Simulation (cont’d.)

- But most complex systems require models that are also complex (to be valid)
  - Must be studied via simulation — evaluate model numerically and collect data to estimate model characteristics
- Example: Manufacturing company considering extending its plant
  - Build it and see if it works out?
  - Simulate current, expanded operations — could also investigate many other issues along the way, quickly and cheaply
1.1 The Nature of Simulation (cont’d.)

• **Some (not all) application areas**
  - Designing and analyzing manufacturing systems
  - Evaluating military weapons systems or their logistics requirements
  - Determining hardware requirements or protocols for communications networks
  - Determining hardware and software requirements for a computer system
  - Designing and operating transportation systems such as airports, freeways, ports, and subways
  - Evaluating designs for service organizations such as call centers, fast-food restaurants, hospitals, and post offices
  - Reengineering of business processes
  - Determining ordering policies for an inventory system
  - Analyzing financial or economic systems

1.1 The Nature of Simulation (cont’d.)

• **Use, popularity of simulation**
  - Several conferences devoted to simulation, notably the Winter Simulation Conference [www.wintersim.org](http://www.wintersim.org)

1.1 The Nature of Simulation (cont’d.)

• **Surveys of use of OR/MS techniques (examples …)**
  - Longitudinal study (1973-1988): Simulation consistently ranked as one of the three most important techniques
  - 1294 papers in *Interfaces* (1997): Simulation was second only to the broad category of “math programming”

1.1 The Nature of Simulation (cont’d.)

• **Impediments to acceptance, use of simulation**
  - Models of large systems are usually very complex
    - But now have better modeling software … more general, flexible, but still (relatively) easy to use
  - Can consume a lot of computer time
    - But now have faster, bigger, cheaper hardware to allow for much better studies than just a few years ago … this trend will continue
    - However, simulation will also continue to push the envelope on computing power in that we ask more and more of our simulation models
  - Impression that simulation is “just programming”
    - There’s a lot more to a simulation study than just “coding” a model in some software and running it to get “the answer”
    - Need careful design and analysis of simulation models – simulation methodology

1.2 SYSTEMS, MODELS, AND SIMULATION

• **System**: A collection of entities (people, parts, messages, machines, servers, …) that act and interact together toward some end (Schmidt and Taylor, 1970)
  - In practice, depends on objectives of study
  - Might limit the boundaries (physical and logical) of the system
  - Judgment call: 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
  - Bank model: Could include number of busy tellers, time of arrival of each customer, etc.
1.2 Systems, Models, and Simulation (cont’d.)

- Types of systems
  - Discrete
    - State variables change instantaneously at separated points in time
    - Bank model: State changes occur only when a customer arrives or departs
  - Continuous
    - State variables change continuously as a function of time
    - Airplane flight: State variables like position, velocity change continuously
- Many systems are partly discrete, partly continuous

1.2 Systems, Models, and Simulation (cont’d.)

- Ways to study a system
  - Simulation is “method of last resort?” Maybe …
  - But with simulation there’s no need (or less need) to “look where the light is”

1.2 Systems, Models, and Simulation (cont’d.)

- 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

1.3 DISCRETE-EVENT SIMULATION

- Discrete-event simulation: Modeling of a system as it evolves over time by a representation where the state variables change instantaneously at separated points in time
  - More precisely, state can change at only a countable number of points in time
  - These points in time are when events occur
- Event: Instantaneous occurrence that may change the state of the system
  - Sometimes get creative about what an “event” is … e.g., end of simulation, make a decision about a system’s operation
- Can in principle be done by hand, but usually done on computer
1.3 Discrete-Event Simulation (cont’d.)

- Example: Single-server queue
  - Estimate expected average delay in queue (line, not service)
  - State variables
    - Status of server (idle, busy) – needed to decide what to do with an arrival
    - Current length of the queue – to know where to store an arrival that must wait in line
    - Time of arrival of each customer now in queue – needed to compute time in queue when service starts
  - Events
    - Arrival of a new customer
    - Service completion (and departure) of a customer
    - Maybe – end-simulation event (a “fake” event) – whether this is an event depends on how simulation terminates (a modeling decision)

1.3.1 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) … described in detail below
  - Fixed-increment time advance (seldom used) … Described in Appendix 1A
    - 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

1.3.1 Time-Advance Mechanisms (cont’d.)

- More on 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

1.3.1 Time-Advance Mechanisms (cont’d.)

- Next-event time advance for the single-server queue
  \[ t_i = \text{time of arrival of } i\text{th customer} \quad (t_0 = 0) \]
  \[ A_i = t_i - t_{i-1} = \text{interarrival time between } (i-1)\text{st and } i\text{th customers} \]
  \[ S_i = \text{service-time requirement of } i\text{th customer} \]
  \[ D_i = \text{delay in queue of } i\text{th customer} \]
  \[ C_i = t_i + D_i + S_i = \text{time } i\text{th customer completes service and departs} \]
  \[ e_j = \text{time of occurrence of the } j\text{th event (of any type)} \quad j = 1, 2, 3, \ldots \]
  - Possible trace of events (detailed narrative in text)
1.3.2 Components and Organization of a Discrete-Event Simulation Model

- Each simulation model must be customized to target system
- But there are several common components, general organization
  - System state – variables to describe state
  - Simulation clock – current value of simulated time
  - Event list – times of future events (as needed)
  - Statistical counters – to accumulate quantities for output
  - Initialization routine – initialize model at time 0
  - Timing routine – determine next event time, type; advance clock
  - Event routines – carry out logic for each event type
  - Library routines – utility routines to generate random variates, etc.
  - Report generator – to summarize, report results at end
  - Main program – ties routines together, executes them in right order

1.3.2 Components and Organization of a Discrete-Event Simulation Model (cont’d.)

- More on entities
  - Objects that compose a simulation model
  - Usually include customers, parts, messages, etc. … may include resources like servers
  - Characterized by data values called attributes
  - For each entity resident in the model there’s a record (row) in a list, with the attributes being the columns

- Approaches to modeling
  - Event-scheduling – as described above, coded in general-purpose language
  - Process – focuses on entities and their “experience,” usually requires special-purpose simulation software

1.4 SIMULATION OF A SINGLE-SERVER QUEUEING SYSTEM

- Will show how to simulate a specific version of the single-server queueing system
- Book contains code in FORTRAN and C … slides will focus only on C version
- Though simple, it contains many features found in all simulation models
1.4.1 Problem Statement

- Recall single-server queueing model
- Assume interarrival times are independent and identically distributed (IID) random variables
- Assume service times are IID, and are independent of interarrival times
- Queue discipline is FIFO
- Start empty and idle at time 0
- First customer arrives after an interarrival time, not at time 0
- Stopping rule: When \( n \)th customer has completed delay in queue (i.e., \( \text{enters} \) service) … \( n \) will be specified as input

1.4.1 Problem Statement (cont’d.)

- Quantities to be estimated
  - Expected average delay in queue (excluding service time) of the \( n \) customers completing their delays
    - Why “expected?”
  - Expected average number of customers in queue (excluding any in service)
    - A continuous-time average
    - Area under \( Q(t) = \text{queue length at time} \ t \), divided by \( T(n) = \text{time} \) simulation ends … see book for justification and details
  - Expected utilization (proportion of time busy) of the server
    - Another continuous-time average
    - Area under \( B(t) = \text{server-busy function} \) (1 if busy, 0 if idle at time \( t \)), divided by \( T(n) \) … justification and details in book
  - Many others are possible (maxima, minima, time or number in system, proportions, quantiles, variances …)
- Important: Discrete-time vs. continuous-time statistics

1.4.2 Intuitive Explanation

- Given (for now) interarrival times (all times are in minutes): 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
- Given service times:
  2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …
- \( n = 6 \) delays in queue desired
- “Hand” simulation:
  - Display system, state variables, clock, event list, statistical counters … all after execution of each event
  - Use above lists of interarrival, service times to “drive” simulation
  - Stop when number of delays hits \( n = 6 \), compute output performance measures

1.4.2 Intuitive Explanation (cont’d.)

- Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
- Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …
1.4.2 Intuitive Explanation (cont’d)

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …
1.4.2 Intuitive Explanation (cont’d)

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Interarrival times: 4, 0.5, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …
1.4.2 Intuitive Explanation (cont’d)

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Interarrival times: 0.4, 0.5, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …
1.4.2 Intuitive Explanation (cont’d)

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Final output performance measures:
- Average delay in queue = 5.7/6 = 0.95 min./cust.
- Time-average number in queue = 9.9/8.6 = 1.15 cus
- Server utilization = 7.7/8.6 = 0.90 (dimensionless)

1.4.3 Program Organization and Logic

- C program to do this model (FORTRAN as well is in book)
  - Event types: 1 for arrival, 2 for departure
  - Modularize for initialization, timing, events, library, report, main
- Changes from hand simulation:
  - Stopping rule: n = 1000 (rather than 6)
  - Interarrival and service times “drawn” from an exponential distribution (mean $\beta = 1$ for interarrivals, 0.5 for service times)

1.4.3 Program Organization and Logic (cont’d.)

- How to “draw” (or generate) an observation (variate) from an exponential distribution?
- Proposal:
  - Assume a perfect random-number generator that generates IID variates from a continuous uniform distribution on [0, 1] … denoted the U(0, 1) distribution … see Chap. 7
  - Algorithm:
    1. Generate a random number $U$
    2. Return $X = -\beta \ln U$
  - Proof that algorithm is correct:

1.4.5 C Program; Simulation Output and Discussion

- Refer to pp. 30, 31, 42-48 in the book (Figures 1.8, 1.9, 1.19-1.27) and the file mm1.c
  - Figure 1.19 – external definitions (at top of file)
  - Figure 1.20 – function main
  - Figure 1.21 – function initialize
  - Figure 1.22 – function timing
  - Figure 1.23 – function arrive (flowchart: Figure 1.8)
  - Figure 1.24 – function depart (flowchart: Figure 1.9)
  - Figure 1.25 – function report
  - Figure 1.26 – function update_time_avg_stats
  - Figure 1.27 – function expon
  - Figure 1.28 – output report mm1.out
  - Are these “the” answers?
  - Steady-state vs. terminating?
  - What about time in queue vs. just time in system?
1.4.7 Alternative Stopping Rules

- Stop simulation at (exactly) time 8 hours (= 480 minutes), rather than whenever $n$ delays in queue are completed
  - Before, final value of simulation clock was a random variable
  - Now, number of delays completed will be a random variable
- Introduce an artificial “end-simulation” event (type 3)
  - Schedule it on initialization
  - Event routine is report generator
  - Be sure to update continuous-time statistics to end
- Changes in C code (everything else is the same)
  - Figure 1.33 – external definitions
  - Figure 1.34 – function main
  - Figure 1.35 – function initialize
  - Figure 1.36 – function report
  - Figure 1.37 – output report mmlalt.out

1.4.8 Determining the Events and Variables

- For complex models, it might not be obvious what the events are
- Event-graph method (Schruben 1983, and subsequent papers) gives formal graph-theoretic method of analyzing event structure
- Can analyze what needs to be initialized, possibility of combining events to simplify model
- Software package (SIGMA) to build, execute a simulation model via event-graph representation

1.5 SIMULATION OF AN INVENTORY SYSTEM;

1.5.1 Problem Statement

- Single-product inventory
- Decide how many items to have in inventory for the next $n = 120$ months; initially (time 0) have 60 items on hand
- Demands against inventory
  - Occur with inter-demand time ~ exponential with mean 0.1 month
  - Demand size = 1, 2, 3, 4 with resp. probabilities 1/6, 1/3, 1/3, 1/6
- Inventory review, reorder – stationary $(s, S)$ policy … at beginning of each month, review inventory level = $I$
  - If $I \geq s$, don’t order ($s$ is an input constant); no ordering cost
  - If $I < s$, order $Z = S - I$ items ($S$ is an input constant, order “up to” $S$); ordering cost = 32 + 3$Z$; delivery lag ~ U(0.5, 1) month

- Demand in excess of current (physical) inventory is backlogged … so (accounting) inventory could be < 0
- Let $I(t)$ be (accounting) inventory level at time $t$ $(+, 0, -)$
  - $I^+(t) = \max \{I(t), 0\}$ = number of items physically on hand at time $t$
  - $I^-(t) = \max \{-I(t), 0\}$ = number of items in backlog at time $t$

- Holding cost: Incur $1 per item per month in (positive) inventory
  - Time-average (per month) holding cost = $\int \frac{I^+(t)dt}{n}$
- Shortage cost: Incur $5 per item per month in backlog
  - Time-average (per month) backlog cost = $\int \frac{I^-(t)dt}{n}$
- Average total cost per month: Add ordering, holding, shortage costs per month
  - Try different $(s, S)$ combinations to try to reduce total cost
1.5.2 Program Organization and Logic

- State variables: Inventory level, amount of an outstanding order, time of the last (most recent) event
- Events:
  1. Arrival of an order from the supplier
  2. Demand for the product
  3. End of the simulation after \( n = 120 \) months
  4. Inventory evaluation (maybe ordering) at beginning of a month
- Random variates needed
  - Interdemand times: exponential, as in queueing model
  - Delivery lags \( \sim U(0.5, 1) \): \( 0.5 + (1 - 0.5)U \), where \( U \sim U(0, 1) \)
  - Demand sizes: Split \([0, 1]\) into subintervals of width \(1/6, 1/3, 1/6\); generate \( U \sim U(0, 1) \); see which subinterval \( U \) falls in; return \( X = 1, 2, 3, \) or \( 4 \), respectively

1.5.4 C Program;

1.5.5 Simulation Output and Discussion

- Refer to pp. 64-66, 73-79 in the book (Figures 1.43-1.46, 1.57-1.67) and the file \texttt{inv.c}
  - Figure 1.57 – external definitions (at top of file)
  - Figure 1.58 – function \texttt{main}
  - Figure 1.59 – function \texttt{initialize}
  - Figure 1.60 – function \texttt{order_arrival} (flowchart: Figure 1.43)
  - Figure 1.61 – function \texttt{demand} (flowchart: Figure 1.44)
  - Figure 1.62 – function \texttt{evaluate} (flowchart: Figure 1.45)
  - Figure 1.63 – function \texttt{report}
  - Figure 1.64 – function \texttt{update_time_avg_stats} (flowchart: Figure 1.46)
  - Figure 1.65 – function \texttt{random_integer}
  - Figure 1.66 – function \texttt{uniform}
  - Figure 1.67 – output report \texttt{inv.out}  
    - Reaction of individual cost components to changes in \( s \) and \( S \) … overall?
    - Uncertainty in output results (this was just one run)?

1.6 ALTERNATIVE APPROACHES TO MODELING AND CODING SIMULATIONS

- Parallel and distributed simulation
  - Various kinds of parallel and distributed architectures
  - Break up a simulation model in some way, run the different parts simultaneously on different parallel processors
  - Different ways to break up model
    - By support functions – random-number generation, variate generation, event-list management, event routines, etc.
    - Decompose the model itself; assign different parts of model to different processors – message-passing to maintain synchronization, or forget synchronization and do “rollback”s if necessary … “virtual time”
  - Web-based simulation
    - Central simulation engine, submit “jobs” over the web
    - Wide-scope parallel/distributed simulation

1.7 STEPS IN A SOUND SIMULATION STUDY
1.8 OTHER TYPES OF SIMULATION

• **Continuous simulation**
  – Typically, solve sets of differential equations numerically over time
  – May involve stochastic elements
  – Some specialized software available; some discrete-event simulation software will do continuous simulation as well

• **Combined discrete-continuous simulation**
  – Continuous variables described by differential equations
  – Discrete events can occur that affect the continuously-changing variables
  – Some discrete-event simulation software will do combined discrete-continuous simulation as well

1.8 Other Types of Simulation (cont’d.)

• **Monte Carlo simulation**
  – No time element (usually)
  – Wide variety of mathematical problems
  – Example: Evaluate a “difficult” integral \( I = \int_a^b g(x) \, dx \)
    - Let \( X \sim U(a, b) \), and let \( Y = (b - a) \, g(X) \)
    - Then \( E(Y) = E[(b - a)g(X)] = \int_a^b g(x) \, dx \)
      - \( = (b - a) \int_a^b g(x) \, f_x(x) \, dx \)
      - \( = (b - a) \int_a^b g(x) \, \frac{1}{b - a} \, dx \)
      - \( = \int_a^b g(x) \, dx \)
      - \( \therefore Y \)’s … this average will be an unbiased estimator of \( I \)

1.9 ADVANTAGES, DISADVANTAGES, AND PITFALLS OF SIMULATION

• Advantages
  – Simulation allows great flexibility in modeling complex systems, so simulation models can be highly valid
  – Easy to compare alternatives
  – Control experimental conditions
  – Can study system with a very long time frame

• Disadvantages
  – Stochastic simulations produce only estimates – with noise
  – Simulation models can be expensive to develop
  – Simulations usually produce large volumes of output – need to summarize, statistically analyze appropriately

• Pitfalls
  – Failure to identify objectives clearly up front
  – In appropriate level of detail (both ways)
  – Inadequate design and analysis of simulation experiments
  – Inadequate education, training