Simulation
CSCI 6337

Xiannong Meng
Department of Computer Science
University of Texas - Pan American

Last revised: July 1998

Chapter 1

Introduction

- A *simulation* is the imitation of the operation of a real-world process or system over time. Steps include
  - Generating an artificial history of a system
  - Observing the behavior of that artificial history
  - Drawing inferences concerning the operating characteristics of the real system
- Use the operation of a bank as an example:
  - Counting how many people come to the bank; how many tellers, how long each customer is in service; etc.
  - Establishing a model and its corresponding computer program.
  - Executing the program, varying parameters (number of tellers, service time, arrival intervals) and observing the behavior of the system.
  - Drawing conclusions: increasing number of tellers; reducing service time; changing queueing strategies; etc.

- The behavior of a system as it evolves over time is studied by developing a simulation *model*.
- A model is a set of *entities* and the *relationship* among them.

  For the bank example: entities would include customers, tellers, and queues. Relations would include customers entering a queue; tellers serving the customer; customers leaving the bank.

- Once developped, a model has to be validated. There are many different ways to validate a model: observation (measurement); analytical model comparison (analysis).

1.1 When is Simulation the Appropriate Tool?

- Simulation enables the study of, and experiment with, the internal interations of a complex, dynamic system, or a subsystem.

  E.g. when setting up a telephone sales department (such as the 1998 World Cup Soccer Game), how many operators are enough to handle the calls? Many factors can play a role here: the number of tickets left, estimated number of people who want the ticket, capacity of the phone lines, how long would it take to service one call, etc.

- Informational, organizational, and environmental changes can be simulated and the effect of these alterations on the model's behavior can be observed.

  E.g. to study the behavior of a web server, we can simulate the client traffic and see how it responds.

- The knowledge gained in designing a simulation model may be of great value toward suggesting improvement in the system under the investigation.

  E.g. before actually building a cache system, one can simulation the various configuration of the cache, study its behavior and find out the optimum solution.

- By changing simulation inputs and observing the resulting outputs, valuable insight may be obtained into which variables are most important and how variabls interact.

  E.g. in studying the performance of a computer network, a number of parameters affect the outcome, cable length, transmission speed, packet size, arrival rate, number of stations, etc. which one is the most important on the delay? It's the ratio of arrival rate and service rate.

- Simulation can be used as pedagogical device to reinforce analytic solution methodologies
- Simulation can be used to experiment with new designs or policies prior to implmentation, so as to prepare for what may happen.
- Simulation can be used to verify analytic solutions.

1.2 Advantages and Disadvantages

Main advantages of simulation include:

- Study the behavior of a system without building it.
- Results are accurate in general, compared to analytical model.
- Help to find un-expected phenomenon, behavior of the system.
- Easy to perform “What-If” analysis.

Main disadvantages of simulation include:

- Expensive to build a simulation model.
- Expensive to conduct simulation.
- Sometimes it is difficult to interpret the simulation results.

1.3 Systems and System Environment

- A *system* is a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose.

- A system is often affected by changes occurring outside the system. Such changes are said to occur in the *system environment*. In modeling a system, it is necessary to decide on the *boundary* between the system and its environment.

  E.g. When studying cache memory using simulation, one has to decide where is the boundary of the system. It can be simply the CPU and cache, or it can include main memory, disk, O.S., compiler, or even user programs.

1.4 Components of a System

- An *entity* is an object of interest in the system. E.g. customers in a bank.
- An *attribute* is a property of an entity. E.g. the checking account balance of the customer.
- An *activity* represents a time period of specified length. Here the time period is emphasized because often the simulation involves time. E.g. deposit money into the checking account at a specified date and time.
- The *state* of a system is defined to be that collection of variables necessary to describe the system at any time, relative to the objectives of the study. E.g. number of busy tellers, number of customers waiting in line.
- An *event* is defined as an instantaneous occurrence that may change the state of the system. E.g. customer arrival, addition of a new teller, customer departure.

1.5 Discrete and Continuous Systems

- A *discrete system* is one in which the state variable(s) change only at a discrete set of points in time. E.g. customers arrive at 3:15, 3:23, 4:01, etc.
- A *continuous system* is one in which the state variable(s) change continuously over time. E.g. the amount of water flow over a dam.

1.6 Model of a System

A *model* is defined as a representation of a system for the purpose of studying the system.

Many times one can't experiment with a real system such as a bank, or a highway system. One has to expriment with a *model* of the real system. A model is often not exactly the same as the real system it presents. Rather, it includes a few (or majority of) key aspects of the real system. It is an *abstraction* of the real system.

1.7 Types of Models

- Static vs. dynamic: A *static* simulation model, sometimes called Monte Carlo simulation, represents a system at particular point in time. A *dynamic* simulation model represents systems as they change over time.
- Deterministic vs. stochastic: A *deterministic* simulation contains no random variable(s). e.g. patients arrvie in a doctor's office at a pre-scheduled time. A *stochastic* simulation involves one or more randome variables as input.
- Discrete vs. continuous: (already discussed).

We are mainly dealing with *discrete-event system simulation*.

1.8 Steps in a Simulation Study

Problem formulation Clearly state the problem.

Setting of objectives and overall project plan How we should approach the problem.

Model conceptualization Establish a reasonable model.

Data collection Collect the data necessary to run the simulation (such as arrival rate, arrival process, service discipline, service rate etc.).

Model translation Conver the model into a programming language.

Verification Verify the model by checking if the program works properly. Use common sense.

Validation Check if the system accurately represent the real system.

Experimental design How many runs? For how long? What kind of input variations?

Production runs and analysis Actual running the simulation, collect and analize the output.

Repeatition Repeat the experiments if necessary.

Document and report Document and report the results.

Chapter 2

Simulation Examples

We will present a few simulation examples of various types here.

2.1 Queueing Systems

- General: See figure 2.1 on page 22 for an illustration of a queueing system.
- Entities: server, waiting line (queue), customers.
- State: number of customers in the system.
- Events: arrival and departure.
- Service flow: figure 2.2 on page 23 and figure 2.3 on page 24.

  Note

  - The flows are organized by the events, one for the departure event and the other for the arrival event.

  - The randomness of the events can be emulated by rolling a dice and recording its face value (so the possible inter-arrival time would be between 1 and 6).

- Tables listed 1) interarrival time 2) service time 3) various clock readings organized by customers 4) chronological ordering of events

Example 2.1: Single-Channel Queue (Single-Server Queue).

- Customers arrive at the check-out counter at random from 1 to 8 minutes apart (interarrival time between 1 and 8 minutes) (Table 2.6 on page 28)
- Service time varies from 1 to 6 minutes with distribution shown in Table 2.7 on page 28.
- Random digit assignment in the tables are used to separate different times. For example, all interarrival time are evenly distributed with a probability of 0.125. When to use 1? When to use 8? This is distincted by the random digit assignment where 1 is used if the RDA is between 1 and 125, 8 is used if RDA is between 876-1000. They have the same length in range (125).
- Running of the simulation: Is one run enough to represent the situation? The answer is 'NO'. Typically, many *random* runs are required to generate sufficient statistical significance.

- Result calculation:

$$Average\ waiting\ time = \frac{total\ time\ customers\ wait\ in\ queue}{total\ number\ of\ customers} = \frac{56}{20} = 2.8\ minutes$$

$$Probability(customer\ has\ to\ wait) = \frac{number\ of\ customers\ who\ wait}{total\ number\ of\ customers} = \frac{13}{20} = 0.65$$

$$Probability(cashier\ idle) = \frac{total\ idle\ time}{total\ run\ time} = \frac{18}{86} = 0.21$$

$$Average\ service\ time = \frac{total\ service\ time}{total\ number\ of\ customers} = \frac{68}{20} = 3.4\ minutes$$

This can be compared with the expected service time from the service time probability distribution

$$Expected\ service\ time = \sum_{s=0}^{\infty} sp(s)$$

which is

$$1 * 0.1 + 2 * 0.2 + 3 * 0.3 + 4 * 0.25 + 5 * 0.1 + 6 * 0.05 = 3.2$$

$$Average\ time\ between\ arrivals = \frac{sum\ of\ all\ times\ between\ arrivals}{number\ of\ arrivals - 1} = \frac{82}{19} = 4.3$$

This can be compared with expected inter-arrival time:

$$E(A) = \frac{a+b}{2} = \frac{1+8}{2} = 4.5$$

The longer the simulation, the closer to their expected values (this is an indication of the quality of the random number generators).

$$Average\ waiting\ time = \frac{total\ time\ customers\ wait\ in\ queue}{total\ number\ of\ customers} = \frac{56}{13} = 4.3$$

$$Average\ time\ a\ customer\ in\ system = \frac{total\ time\ customers\ in\ queue}{total\ number\ of\ customers} = \frac{124}{20} = 6.2$$

Note: typo on page 30, third equation: it should read “total idle time ...”, rather than “total run time ...”.

Example 2.2: The Able Baker carhop problem

The first example involves one server (one cashier). This example will show the case for two servers, one serves faster than the other.

If both servers are idle when a new customer comes in, Able gets the work.

A few statistics:

- The total elapse time is 62 minutes.
- Able was busy 90% of the time.
- Baker was busy 69% of the time.
- Nine out of the 26 arrivals (about 35%) had to wait. Average waiting time for all customers is about 0.42 minutes.
- Those nine, who actually waited, only waited an average of 1.22 minutes.
- Adding another server can probably reduce the waiting time to zero, but it will not be very economic.

2.2 Simulation of Inventory Systems

Some amount M of goods is in stock to begin with. As sales continue, the stock decreases. At the pre-defined interval, N, the stock level is checked, and the goods is re-ordered.

- It is possible that the inventory becomes negative, meaning the goods is in shortage.
- The *lead time*, which is the time between the issurance of re-order and the arrival of the goods, can be considered as zero in many cases.

Example 2.3: The newspaper seller's problem

- The paper seller buys the papers for 33 cents each and sells for 50 cents each.
- The papers not sold at the end of the day are sold as scrap for 5 cents each.
- Newspapers can be purchased in bundles of 10. Thus the paper seller can buy 40, 50, 60, and so on. In the simulation shown in Table 2.18 the case of purchasing 70 newspapers is demonstrated.
- There are three types of newsdays, "good", "fair", and "poor" with probabilities of 0.35, 0.45 and 0.20.
- The demand distribution is listed in Table 2.15 on page 37.
- Profit is calculated as

  profit = revenue - cost - lost profit from excess demand + salvage
- Table 2.16, 2.17 and 2.18 show the simulation tables.

Example 2.4: Simulation of an (M,N) inventory system.

- M is the maximum inventory level, assume it is 11 units.
- N is the length of review period, assume it is 5 days.
- The initial inventory is 3 units, and an initial order of 8 units is scheduled to arrive in 2 days. This is the initial settting of the simulation.
- Table 2.21 on page 41 shows the detail.

2.3 Other Examples of Simualtion

Example 2.5: Reliability Problem.

- A machine has three different bearings that may fail in service.
- Bearing-life distribution is in Table 2.22.
- When a bearing fails, the mill stops, a repair-person is called, and a new bearing is installed.
- Cost for the bearing failures:

downtime	\$5 / minute
repair-person	\$15/ hour
time to change the bearing	20 minutes for one bearing,
	30 minutes for two, 40 minutes for three
bearing cost	\$16/each

- Table 2.24 shows the result of replacing only the broken bearings.
- Table 2.25 shows the result of the new policy: replace all three bearings if any one is broken.
- The net result is that the saving of the new policy is about \$865 a year.

Example 2.6: Random normal numbers.

- A squadron of bombers attempt to destroy an anmmunition depot (see Figure 2.8 on page 44)
- If a bomb lands in an area of 600 meters (horizontal) by 300 meters (vertical) away from a specified center, it is a hit. Otherwise it is a miss.
- We use two numbers to indicate the position that a bomb hits, X for horizontal and Y for vertical. Two separate normally distributed random numbers are generated, one for X and the other for Y.
- The result is shown in Table 2.26 on page 47.
- This type of simulations doesn't involve time. They are called Monte-Carlo, or static simulation.

Chapter 3

General Principles

This chapter develops a common framework for the modeling of complex systems using discrete-event simulation. Discuss basic building blocks of discrete-event simulation: entities and attributes, activites and events, states.

3.1 Concepts and Definitions

System: A collection of entities (e.g. people and machines) that interact together over time to accomplish one or more goals.

Model: An abstract representation of a system, usually containing structural, logical or mathematical relationships which describe a system in terms of state, entities and their attributes, sets, processes, events, activities, and delays.

E.g. a bank operation can be modeled as a queueing system: tellers being the server(s), customers being the jobs in the queue. We ignore all other features of a bank in this model (doors, receptionist, waiting areas, etc.)

System state: A collection of variables that contain all the information necessary to describe the system at any time.

E.g. In the Able-Baker carhop example, we may define the system state as (Able-busy, Baker-busy, number-of-cars-waiting-in-line).

Entity: Any object or component in the system which requires explicit representation in the model.

E.g. a server, a job on an assembly line, a machine.

Attributes: The properties of a given entity.

E.g. the priority of a waiting customer, the routing distribution of a job through a workshop.

List: A collection of associated entities, ordered in some logical fashion.

E.g. when simulating a computer network with multiple computers, packets waiting to be transmitted at each computer form a list ordered by the time of their generation. There will be multiple instances of this kind of lists.

Event: An instantaneous occurrence that changes the state of a system (such as an arrival or departure of a customer).

Event notice: A record of an event to occur at the current or some future time, along with any associated information necessary to execute the event. At a minimum, the record includes the event type and the event time.

E.g. when simulating the operation of an airport, we may have two types of events, *take-off* and *landing*. With these two events, a possible event notice would take the following form.

- Event type (e.g. landing or take-off)
- Event time (e.g. 134)
- Flight number
- Aircraft type (e.g. Boeing 737-200, DC-10)
- Number of passengers on board (e.g. 125)
- Pointer to other flight information
- Pointer to the aircraft specification
- Pointer to the crew information

Event list: A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL).

Activity: A duration of time of spcified length (e.g. a service time or an inter-arrival time), which is known when it begins (although it may be defined in terms of statistical distribution).

Note that the term *time* here is not necessarily a reading of the clock, rather it is a process (e.g. life-time of a bearing).

E.g. take-off time: an aircraft will complete its take-off in three minutes after it starts its engine.

E.g. a customer starts to be serviced at a barber shop, it may take the barber a random amount of time to finish.

Delay: A duration of time of unspecified indefinite length, which is not known until it ends.

E.g. a customer's delay in a last in, first out waiting queue which, when it begins, depends on future arrivals (e.g. procedure call stack).

Clock: A variable representing simulated time. `CLOCK` is used in the textbook examples.

The clock variable can be centralized or distributed.

3.2 Execution Mechanism of Discrete-Event Driven Simulation

With the basic concepts discussed, how is a typical discrete-event driven simulation executed? We will describe this process by using an example: airport simulation (take-off and landing) with one run way.

1. Possible events: landing-requst, landing, landing-complete, take-off-request, take-off, take-off-complete, idle.
2. A initial state of the simulation is set.
   - Set runway to be idle.
   - No landing or take-off is taking place.
   - The simulation clcok is zero.
   - The first landing-request is scheduled at time 3; the first take-off-request is scheduled at time 5.
3. At this memont, the FEL has two event notices, landing-request at 3 and take-off-request at 5.
4. Both landing and take-off take 3 minutes to complete.
5. Take the first event notice off the FEL, process it (a landing-request event). Processing event notice typically involves programming activities tailed towards the applications. For example,
   - Set run way to be busy so no other landing, take-off can take place.
   - Generate next event notice of the same type (landing-request). The time of the next event is determined either by a fixed interval, or drawn from a random number generator. Assume the time is 4. This new event notice is inserted into the FEL, before the original second event notice of take-off at 5!
   - Generate a landing-complete event notice at time 6. Insert it into the FEL.
   - Collect information: how many passengers on board, flight number, etc.
6. Take the next event notice. This time, another landing-request at time 4. But the runway is busy. So we have to put this event notice into the waiting queue for the runway. (Note that we don't put this event notice back to the FEL in this case.)
7. Take the next event notice. This is a take-off-request event at time 5. The runway is still busy. Put it into the waiting queue for the runway.
8. Take the next event notice. This is a landing-complete event at time 6. Processing the event:
   - Set the runway to be free.

- Updating statistics (e.g. another landing completed).
- Check if there is any airplanes waiting in the waiting queue for the runway. If there is, take the event notice off the waiting queue and process it. In this example, the first event notice in the waiting queue is a landing-request, the scheduled event time is 4. The actual event time is 6. When processing this event, a landing-complete event at time 9 is inserted into the FEL.

9. This process is repeated until a pre-defined condition is met such as total simulation time is reached, or the total number of landing, take-off is reached.
10. Every time an event notice is processed, the CLOCK value is set to be the value of the event time. This is called the simulation clock, or simulation time.
11. The method of generating future event when processing a current event of the same type is called *bootstrapping*.
12. The events that appear in FEL are called *primary events*. Others are called *conditional events* such as the event *lading* and *take-off*, which do not appear in FEL.
13. Another possible way of generating primary events is alternating a state. For example, the airport might have to be shut down in a random fashion. Otherwise it is in normal operating mode. To simulate this fact, one can schedule an *end-of-normal* event in some future time. When that time is reached, the airport becomes *shut-down*. When processing *end-of-normal* event, an *end-of-shut-down* event has to be generated in the future.

3.3 World Views

How to describe a simulation, or from what point of view to observe a simulation? There are a few popular modes. These are called *world views*.

event-scheduleing world view: We view the simulation as a sequence of events scheduled according to their event time. The simulation is proceeded by a sequence of snap-shots of the system. Each snap-shot is triggered by a event from the event list.

Only one sanp-shot (the current one) is kept in computer memory. A new snap-shot can be derived only from the previous snap-shot, newly generated random variable values, and the event logic. Past snap-shots should be ignored when advancing the clock. The current snap-shot must contain all information necessary to continue the simulation.

We will see more examples later.

process-interaction world view: In process-interaction world view, the simulation is considered as a collection of interactions among processes. It is similar to the object-oriented programming paradigm. Processes interact with each other by messages.

See Figure 3.4 on page 69 for an example. From the view point that two processes interact with each other.

Often specialized simulation package can support this view. These simulation packages take care of the time advancing issues for the programmers. Programming in general purpose high level language is difficult to use this process-interaction world view because it will be too complicated for programmers to specify all the details.

activity-scanning world view: With activity-scanning approach, a modeler concentrates on the activities of a model and those conditions that allow an activity to begin. At each clock advance, the conditions for each activity are checked and if the conditions are true, the corresponding activity begins.

See Figure 3.4 on page 69 for an example. From the view point that activities and conditions.

3.4 Other Examples

Example 3.1: Able and Baker, revisited.

- System state:
  - $L_Q(t)$: the number of cars waiting to be served at time t.
  - $L_A(t)$: a boolean variable indicating Able being idle or busy at the time.
  - $L_B(t)$: a boolean variable indicating Baker being idle or busy at the time.
- Entities: cars and the two servers.
- Events:
  - arrival event
  - service complete by Able event.
  - service complete by Baker event.
- Activities:
  - interarrival time
  - Able's service time
  - Baker's service time
- Delay: a customer's wait in the queue until Able or Baker is free.

Example 3.3: Single-channel queue (Supermarket check-out counter).

In conducting an event-scheduling simulation, a simulation table is used to record the successive system snap-shots as time advances.

The simulation table is Table 3.1 on page 72.

System state: (LQ(t), LS(t)) where LQ(t) is the number of customers in waiting line, and LS(t) is the number of customers in service at time t.

Entities: Server and customers.

Events:

- Arrival (A)
- Departure (D)
- Stopping event (E), scheduled to occur at time 60.

Event notices:

- (A,t) representing an arrival event to occur at future time t.
- (D,t) representing a departure event to occur at future time t.
- (E,60) representing the stopping event to occur at future time 60.

Activities:

- Interarrival time, defined in Table 2.6 page 28.
- Service time, defined in Table 2.7 page 28.

Example 3.4: The check-out counter simulation, continued.

Further from Example 3.3, we want to collect some statistics, mean response time and mean proportion of customers who spend 4 or more minutes in the system (time in the system includes waiting time and service time).

- mean response time $= \frac{\sum_{i=0}^{i=n} depart_i - arrive_i}{n}$
- mean proportion of the customers who spend 4 or or minutes $= \frac{number\ of\ cusotmers\ who\ waits\ 4\ minutes\ or\ more}{total\ number\ of\ customers}$

Example 3.5: The dump truck problem.

- Six dump trucks haul coal from the entrance of small mine to the railroad.
- Each truck is loaded by one of the two loaders.
- After loading, a truck moves immediately to the scale to be weighed as soon as the scale is available.
- The two loaders and the scal have a first-come-first-server queue.
- The actual time for loading and weighing is negligible.
- Aftger being weighed, the truck drives off and will come back to the same line, to get smore coal. This process is repeated.

The model has following components.

- System state: [LQ(t), L(t), WQ(t), W(t)]
  - LQ(t) = number of trucks in loader queue
  - L(t) = number of trucks in loader (0, 1, 2)
  - WQ(t) = number of trucks in scale's waiting queue.
  - W(t) = number of trucks being weighed (0, 1, or).
- Event notices
  - (ALQ, t, DTi) dump truck i arrives at loader queue
  - (EL, t, DTi) dump truck i end ends loading at EL being loaded.
  - (EW, t, DTi) dump truck i ends weighing at time t.
- Entities: The six dump trucks (DT1, DT2, ..., DT6)
- Lists: loader queue and scale queue, both first-in-first-out.
- Activites: Loading time, weighing time, and travel time.
- Delay: Delay at loader queue, and the scale.

See Table 3.6 on page 77 for an illustration.

Chapter 4

Parallel Discrete Event Simulation

4.1 Basics

- Parallel discrete event simulation (PDES) refers to the execution of a single discrete event simulation program on a parallel computer.
- A discrete event simulation model assumes the system being simulated only changes state at a discrete points in simulated time.
- The simulation model jumps from one state to another upon the occurrence of an *event*.
- In a real world system model, many things (events) can occur at the about same time, yet few take place at the exact same moment. Also, they don't have a regular interval in between occurrances.
- *Asynchronous* systems where events are not synchronized by a global clock.
- A possible mechanism of PDES is to use *lock-step* execution using a global simulation clock among many processors. At each step of simulated time, event lists on different processors are checked and the events due in time are executed.
- This approach performs very poorly because very few events would have the exact same time.

- *Concurrent* execution of events at *different* points in simulated time is required! This introduces interesting synchronization problems that are at the heart of the PDES problem.

- Sequential simulations typically utilize three data structures:

  1. the *state variables* that describe the state of the system,
  2. an *event list* containing all pending events that have been scheduled, but have not yet taken effect, and
  3. a *global clock* variable to denote how far the simulation has progressed.

- In this execution paradigm, it is crucial that one always select the smallest timestamped event from the event list as the one to be processed next.

- If an event with larger timestamp were executed before a smaller one that would schedule this larger timestamped event, an logic error would occur. We call this type of errors *causality* errors.

  Example: if a customer's departure event is processed before its arrival event, a causality error occured.

- The greatest opportunity in parallel simulation is to process events concurrently on different processors.

- A typical strategy is to map each physical process to a logical process (LP) and each LP proceeds simulation on its own pace.

  Example: if we are to simulate a gas station with two independent attendants, they form naturally two LPs, each of which can execute simulation of its own.

- One can ensure that no causality errors occur if one adheres to the following constraint:

  Local Causality Constraint A discrete event simulation, consisting of logical processes (LPs) that interact exclusively by exchanging timestamped messages, obeys the local causality constraint if and only if each LP processes events in non-decreasing timestamp order.

  The above LCC essentially says if events are processed in non-decreasing timestamp order, then we say it obeys the causality constraint; if events are not processed in non-decreasing timestamp order, then we say it does not obey the causality constraint.

- Adherence to this constraint is sufficient, though not always necessary, to guarantee that no causality errors occur. In another words, violating causality constraint may not always result in simulation error. This is because two events within a single LP may be independent of each other, in which case processing them out of timestamp sequence does not lead to causality error.

  Example: a supermarket has a service desk and a number of check-out lines. The customers who go through service desk can be considered independent of those who go through check-out lines. If we process them out of timestamp order, it will not lead to causality error.

  On the other hand, if in the same check-out line for a single customer, if we process the *bagging* event before the *check-out* event, a causality error has occured.

- The challenge in PDES is to execute LPs concurrently and lead to correct simulation results.

- PDES mechanisms can be divided into two categories: *conservative* and *optimistic*.

  Conservative approaches strictly *avoid* the possibility of any causality error. Typically these approaches reply on some strategy to determine if it safe to process an event.

  Optimistic approaches use a *detect and recover* strategy: causality errors are allowed, but detected, and a *rollback* mechanism is invoked to recover the errors.

4.2 Conservative Approach

- The key to conservative PDES is to make sure no causality error will occur before processing an event. Different strategies exist.

- Statically specify the links that indicate which processes may communicate with which other processes. An LP can send messages only to specified LPs.

  E.g.: when simulating the operation of a number of airports with airplanes taking-off and landing, we may speicify that airplanes take off from A can only land on B and C. This way we have a fixed link.

- Messages arriving on each incoming link are stored in FIFO order for the LP to process. Each of the incoming links maintains a clock that is equal to either the timestamp of the message at the head of the queue if the queue is not empty, or the timestamp of the last received message if the queue is empty.

- The LP repeatedly selects the link with the samllest clock and, if there is a message in that link's queue, processes it.
- If the selected queue is empty, the process blocks.
- The above protocol guarantees that each process will only process events in non-decreasing timestamp order, thereby ensuring adherence to the local causality constraint.
- Problems: if a cycle of empty queues arises that all have small clock value, each process in the cycle must block, and the simulation deadlocks. (See Figure 2 on page 34 of Fujimoto's 1990 CACM paper).
- To solve this problem, a *null* message is utilized. An LP sends messages to all of its out-going links in a round-robin fashion. If at a time the LP doesn't have a message for on particular out-going link, a null message with timestamp T_{null} is sent to that link. Doing so guarantees that the receiving LP will not have to block, thus deadlock is avoided.
- The null message from LP_A with timestamp T_{null} essentially tells the reciving process that LP_A will not send events in the future to other LPs with timestamp smaller than T_{null}.
- How to compute T_{null}? T_{null} is the minimum of all incoming links timestamp and first event's timestamp on LP's own event list.
- This protocol works correctly. However, it generates large number of null messages, thus waste of processing time.
- Large amount of work has been done to improve the performance of conservative approach to PDES.

4.3 Optimistic Mechanisms

- Optimistic mechanism allow all LPs proceed because it is possible that causality error might not occur.

- If at a later stage an causality error is detected, the recover process takes place, undoing the effects of the events that have been processed (rollback).
- The event causing rollback is called a *straggler*.
- An event may do two things that have to be rolled back:
  1. it may modify the state of the logical process;
  2. it may send event messages to other processes
- Rolling back the state is accomplished by periodically saving the process's state, and restoring an old state vector on rollback. The state has to be rolled back to a simulation time that is equal or smaller than the time of the *straggler*.
- "Unsending" a previously sent message is accomplished by sending a negative message or an *anti-message* that annihilates the original message when it reaches its destination.
- If a process receives an anti-message that corresponds to a positive message that has already been processed, this anti-message becomes a straggler, causing rollback on this LP.
- Recursively repeating this procedure allows all the effect of erroneous computation to eventually be canceled. It can be proved that this process converges, and it always makes progress under certain conditions.
- The fundation of the optimistic approach is *virtual time*, which is one of the most important concepts in distributed computing.
  - A *virtual time system* is a distributed system executing in coordination with an imaginary *virtual clock* that ticks *virtual time*.
  - Virtual time itself is a global, one-dimentional, temporary coordinate system imposed on a distributed computation.
    * It is used to measure computational progress and to define synchronization.
    * It may or may not have a connection with real time.
    * It is a real positive value, totally ordered by $<$.
  - Virtual time systems are subject to two fundamental rules.
    1. The virtual send time of each message must be less than its virtual receive time.

2. The virtual time of each event in a process must be less than the virtual time of the next event at that process.

These rules are exactly what Lamport's Clock Conditions (the well-known Happens-Before relation).

– The major constraint on the implementation of virtual time can be stated: *If an event A causes event B, then the exeuction of A and B must be scheduled in real time so that A is completed before B starts.*

  This implies that if A does not cause B, B could be executed before A in real time, even if the logical time of B is after that of A.

– One implementation of virtual time is the Time Warp system.

  ∗ For correct implementation of virtual time, it is necessary and sufficient that *at each process* messages are handled in timestamp order.

  ∗ Structure of the run time representation.

    · A unique *process name.*

    · A *local virtual clock* (LVT) which has to be compatiable with the *global virtual time* (GVT), but it doesn't have to have the same value as GVT.

    · A *state* which is a collection of variables.

    · A *state queue* containing saved copies of the process's recent states.

    · An *input queue* containting all recently arrived messages sorted in order of virtual receive time.

    · An *output queue* containing all negative copies of the messages this process has recently sent, kept in virtual send time order. They will be used in case of rollback.

  ∗ See Figure 1 and Figure 2 in Jefferson's 1985 paper for illustration.

  ∗ Global virtual time: GVT at real time r is the minimum of (1) all local virtual times in all local virtual clock at time r, and (2) of the virtual send times of all messages that are in transition.

    · GVT never decreases.

    · GVT can serve as a floor for the virtual times to which any process can ever again roll back.

    · GVT can be viewed as a moving *commitment horizon*: any event with virtual time less than GVT cannot be rolled back and may be removed from the system.
    Thus the name *time warp.*

        · If (1) every event completes normally, and (2) messages are delivered reliably, and (3) the scheduler does not indefinitely postpone execution of the farthest-behind process, and (4) there is sufficient memory, then *GVT must eventually increase.*

– Examples of virtual time system include distributed discrete event simulation, distrbuted database concurrency control, virtual circuit communicaiton.

– Virtual time has an anlogy to virtual memory in memory management.

    ∗ The virtual address space of a page is its spatial coordinate; the virtual time of an event is its temporal coordinate.
    ∗ A page resident in main memory at time t is analogous to an event with a virtual time in the future of process x, where the page may be accessed in the future, the event will be processed in the future.
    ∗ A page out of memory at time t is analogous to an event in the present or past of process x.
    ∗ Accessing a page in memory is relatively inexpensive, but accessing a page out of memory at time t is very expensive (*page fault*); similarily sending a message that arrives in the virtual future of the receiving process is relatively inexpensive, while sending a message into its virtual past causes a very expensive *time fault*, that is, rollback.
    ∗ Under a virtual memory system, it is only cost-effective in time to execute programs that obey the spatial locality principle, so that most memory accesses are to pages already resident in memory, and page faults are relatively rare. Likewise, under a virtual time system, it is only cost-effective to run programs that obey temporal locality principle, that is, most messages arrive in the virtual future of the destination processes so that time faults are relatively rare.
    ∗ The term "memory mapping" refers to the translation of virtual addresses to real address. We could use *time mapping* to refer to the mapping of virtual times to real times. The same virtual address may be mapped to different physical address in memory at different time, and similarily, the same virtual time may be mapped to (scheduled at) different real times at different places.
    ∗ The only acceptable memory maps are the one-to-one functions because they preserve *distinctness*, mapping distinct virtual addresses (pages) to distinct real addresses (frames). At any given moment, some virtual

addresses may be unmapped because they refer to pages not in memory. Similarly the only acceptable time maps are the strictly increasing function because they preserve *order*, mapping distinct virtual times into disctinct real times. At any given place, some events may be unmapped (not yet scheduled) because they are in the local future.

    * For a process running under virtual memory, the page fault rate can usually be reduced by increasing the number of pages it is permitted to have in main memory. Similarily the time fault rate of a process running under a virtual time can usually be reduced by increasing the number of events that are in its near future. Essentially, more interaction among LPs reduces the chances of time fault. But it may slow down the simulation progress.
    * If a process can have enough pages in memory, its page fault rate can be reduced to zero; in general, this is undesirable because it leads to inefficient use of main memory. Similarily, if a process has sufficiently large number of events to process in a short period of time, its time fault rate can be reduced to zero; this too is undesirable because this process then becomes bottleneck, holding back the progress of GVT, leading to inefficient use of real time (other processes have to wait).

- GVT calculation. There are many different ways to compute GVT.
  - Centralized calculation: Each LP sends its LVT to a centralized manager, the minimum of these is the calculated GVT.
  - Distributed calculation: LPs broadcast their LVT periodically. Each LP calculates GVT to be the minimum of all LVTs (need non-blocking read).
- Rollback distance vs. forwarding distance (a model of random walk)
  - Rollback converges (partial ordering of the events, cases for events that have the same time stamp, and different time stamp)
- Memory management, all issues are memory vs. speed
  - Interleaved state saving (How often?) Instead of saving every single input/output events, interleaved state saving strategy only saves a snap shot of the state at certain time. When a rollback is needed, the simulation is rolled back to the point where the state is saved.
  - Fossil collection: events that have a clock value smaller than GVT will never be used again, removing them can recover some memory

    * lazy: don't collect until running our memory
    * aggressive: collect at a fixed interval or upon a certain event
  - Return un-processed messages

    When memory is running low, one can return un-processed input events back to the sender to free up some memory. The ones with highest time stamps should be returned first because they are less likely to have effects on the system.
  - Gafni's protocol: instead of always returning un-processed message, reclaiming memory according to the nature of the activity.
    * If an output event is the cause, a corresponding anti-event is sent. The local simulation is restored to a previous state.
    * If an input event is the cause, do conventional return-message.
    * If a state saving is the cause, discard that set of state variables. They can be re-computed later if needed.
    * Artificial rollback: with the absence of straggler, rollback is called artificially to reclaim memory.

- Communications in PDES
  - Communication overhead
  - Difference between parallel simulation (shared physical memory) and distributed simulation (message passing)
  - In distributed simulation: less communication would reduce total delay, may cause more rollback
  - The relative value counts: i.e. the ratio of computing time vs. communication time

- Performance models:
  - Chen's work: unify the performance model for memory effect on message distribution, rollback, GVT advances in shared memory environment using a Markov's process model
  - Gupta, Akyildiz, and Fujimoto: performance model for homogeneous and heterogeneous multiprocessor, shared memory environment
  - Kleinrock and Nicol have separately developed some bounds for the performance of Time Warp system

– Lubachevsky, Weiss and Shwartz's work: the average time τ to complete the simulation of a system with N nodes and R events on a p-processor PRAM satisfies

$$\tau \;=\; O\left(\frac{R}{p} + \frac{R}{N} log\ p\right)$$

4.4 References

- L. Chen, "Performance Analysis and Improvement of Parallel Simulation", Ph.D. Dissertation, Georgia Institute of Technology, May 1993
- A. Gupta, I.F. Akyildiz, R.M. Fujimoto, "Performance analysis of Time Warp with multiple homogeneous processors", *IEEE Trans. on Softw. Eng.* (17), 1991
- A. Gupta, I.F. Akyildiz, R.M. Fujimoto, "Performance analysis of Time Warp with multiple heterogeneous processors" 1993
- L. Kleinrock, "On distributed systems performance", *Computer Networks and ISDN Journal*, (20) 1990
- D.M. Nicol, "Performance bounds on parallel self-initiating discrete event simulations", *ACM Trans. on Model. and Comput. Simul.*, 1991
- B. Lubachevsky, A. Weiss and A. Shwartz, "An analysis of rollback-based simulation", *ACM Trans. on Model. and Comput. Simul.*, April, 1991
- C.D. Carothers and R.M. Fujimoto, "Effect of Communication Overhead on Time Warp Performance: An Experimental Study", *Proceedings of 8th Workshop in Parallel and Distributed Simulations*, 1994
- K.M. Chandy and J. Misra, "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs", *IEEE Trans. on Software Engineering*, SE-5, 5, September 1979.
- David R. Jefferson, "Virtual Time", *ACM Transactions on Programming Languages and Systems*, 7(3), July 1985.
- J. Misra, "Distributed Discrete-Event Simulation", *ACM Computing Surveys*, Vol. 18, No. 1, pp. 39-65, March 1986

- D.R. Jefferson et.al., “Distributed Simulation and the Time Warp Operating System”, *Proceedings of the 11th ACM Symposium on Operating Systems Principles*, November, 1987
- Richard Fujimoto, “Parallel Discrete Event Simulation”, *Communications of ACM*, 33(10), October 1990.
- David Nicol and Richard Fujimoto, “Parallel Simulation Today”, *College of William & Mary Technical Report*, 1992
- A. Ferscha and S.K. Tripathi, “Parallel and Distributed Simulation of Discrete Event Systems”, Technical Report, Dept. of Comp. Sci., Univ. of Maryland, CS-TR-3336, August 1994

Chapter 5

Statistical Model of Simulation

- In this chapter, we discuss some of the most commonly used statistical models.
- Motivation:
  - So far we used only the uniformly distributed random numbers. They only fit a fraction of the real applications.
  - We need other kind of statistical models to describe various types of applications.
  - These models can be used in other situations than simulation. But we discuss them in the context of simulation.
- There are two major categories of random numbers:discrete and continueous.
- The discrete distributions we will discuss inculde:
  1. Bernoulli distribution
  2. Binomial distribution
  3. Geometric distribution
  4. Poisson distribution
- Contineous distributions include:
  1. Uniform distribution
  2. Exponential distribution
  3. Gamma distribution
  4. Erlang distribution
  5. Normal distribution
  6. Weibull distribution

7. Triangle distribution

- Note that here we discuss random variables that are *discretely* or *contineously* distributed. This doesn't have a direct connection to *discrete* simulation vs. *contineous* simulation where the concept is how the simulation clock is ticked.
- We will talk about some of the basics of probabilities first. Then we will discuss various distributions. Then we will study the Poisson process, a very important, yet relatively simple distribution.

5.1 Review of Terminology and Concepts

- *Random variable*: A variable that assumes values in a irregular pattern (or no particular pattern).
- *Discrete random variables*: Let X be a random variable. If the number of possible values of X is finite, or countably infinite, X is called a discrete random variable.

  Let x_i be all possible values of X, and $p(x_i) = P(X = x_i)$ be the probability that $X = x_i$, then $p(x_i)$ must meet the following conditions.

  1. $p(x_i) \geq 0$ for all i.
  2. $\sum_{i=1}^{\infty} p(x_i) = 1$.

  Examples: example 6.1 and 6.2 on p. 186
- *Contineous random variables*: If the values of X is an interval or a collection of intervals, then X is called a contineous random variable.

  For contineous random variable, its probability is represented as

  $$P(a \leq X \leq b) = \int_a^b f(x)dx$$

  The function $f(x)$ is called the probability density function (pdf) of the random variable X, which has to meet the following condition.

  1. $f(x) \geq 0$ for all x.
  2. $\int f(x)dx = 1$.
  3. $f(x) = 0$ if x is not in R_X.

  Example 6.3 on page 187.
- *Cumulative distribution function.* The cumulative distribution function (cdf), denoted by $F(x)$, measures the probability that the random variable X assumes a value less than or equal to x, $F(x) = P(X \leq x)$.

- If X is discrete, then $F(x) = \sum_{\text{all } x_i \le x} p(x_i)$
- If X is contineous, the $F(x) = \int_{-\infty}^{x} f(t)dt$

Some propertities of cdf include:

1. F is a non-decreasing function. If $a < b$ then $F(a) \le F(b)$.
2. $lim_{x \to \infty} F(x) = 1$.
3. $lim_{x \to -\infty} F(x) = 0$.

Example: 6.4, 6.5 on page 189.

- *Expectation and variance.* Expectation essentially is the expected value of a random variable. Variance is a measure how a random variable varies from its expected value.
  - For discrete random variables
  $$E(X) = \sum_{\text{all } i} x_i p(x_i)$$
  - For continueous random variables
  $$E(X) = \int_{-\infty}^{\infty} x f(x) dx$$
  - For discrete or continueous random variables, its variance is
  $$V(X) = E[(X - E[X])^2]$$
  which has an identity
  $$V(X) = E(X^2) - [E(X)]^2$$
  - A more frequently used practical measure is *standard deviation* of a random variable, which is expressed as the same units as that of expectation.
  $$\sigma = \sqrt{V(X)}$$

  Examples: 6.6, 6.7 on page 191.

- *The mode.* The mode is used to describe most frequently occured values in discreate random variable, or the maximum value of a continueous random variable.

5.2 Useful Statistical Models

1. *Queueing system.* This is one of the most often used models. We will study queueing system in great detail later. Queueing system can have a lot of variations that can be used to model many real life situation. We have seen a few examples in earlier chapters.

   Queueing systems often include random variables such as service time at various server(s); inter-arrival time for various customer streams; routing distribution of customers.

   Example 6.8, 6.9 on page 193. Emphasize on the case where empirical data can produce histogram which shows the trend, often close to a mathematical model.

2. *Inventory system.* This is another model used to simulation storage problems, inventory problems. A typical inventory has three random variables: the number of units demanded per order or per time period; the time between demands; the lead time.
3. *Reliability and maintainability.* Time to failure of a component, of a system.
4. *Limited data.* When complete data is not avaiable, three distributions can be used to model the system, uniform, triangular, and beta distribution.
5. *Other distributions.* A few other distributions can be used to model systems that are not covered above: Bernoulli and binomial distributions are for discrete system; hyperexponential may be useful in other instances.

5.3 Discrete Random Variables

Here we are going to study a few discrete random variable distributions.

1. *Bernoulli trials and the Bernoulli distribution.*
   - A Bernoulli trail is an experiment with result of success or failure.
   - We can use a random variable to model this phenomena. Let $X_j = 1$ if it is a success, $X_j = 0$ if it is a failure.
   - A consecutive n Bernoulli trails are called a Bernoulli process if
     * the trails are independent of each other;
     * each trail has only two possible outcomes (success or failure, true or false, etc.); and
     * the probability of a success remains constant
   - The following relations hold.
     (a) $p(x_1, x_2, ..., x_n) = p_1(x_1) * p_2(x_2) * ... * p_n(x_n)$ which means the probability of the result of a sequence of events is equal to the product of the probabilities of each event.
     (b) Because the events are independent and the probability remains the same,
     $$p_j(x_j) = p(x_j) = \begin{cases} p & x_j = 1, j = 1, 2, ..., n \\ 1 - p = q & x_j = 0, j = 1, 2, ..., n \end{cases}$$
   - Note that the "location" of the p_is don't matter. It is the count of p_is that is important.
   - Examples of Bernoulli trails include: a conscutive throwing of a "fair" coin, counting heads and tails; a pass or fail test on a sequence of a particular components of the "same" quality; and others of the similar type.
   - For one trial, the distribution above is called the Bernoulli distribution. The mean and the variance is as follows.
   $$E(X_j) = 0 * q + 1 * p = p$$
   $$V(X_j) = E[X^2] - (E[X])^2 = [0^2 * q + 1^2 * p] - p^2 = p(1 - p)$$

2. *Binomial distribution.*
   - The number of successes in n Bernoulli trials is a random variable, X.
   - What is the probability that m out of n trials are success?

$$p_{m,n} = p^m q^{n-m}$$

   - There are

$$\left(\begin{array}{c} n \\ x \end{array} \right) = \frac{n!}{x!(n-x)!}$$

   - So the total probability of m successes out of n trials is given by

$$p(x) = \begin{cases} \left(\begin{array}{c} n \\ x \end{array} \right) p^x q^{n-x} & x = 0, 1, 2, ..., n \\ 0 & \text{otherwise} \end{cases}$$

   - Mean and variance: consider the binomial distribution as the sum of n independent Bernoulli trials. Thus

$$E[X] = p + p + ... + p = np$$

$$V(X) = pq + pq + ... + pq = npq$$

   - Example 6.10 on page 198

3. *Geometric distribution.*
   - The number of trials needed in a Bernoulli trial to achieve the first success is a random variable that follows the *geometric distribution.*
   - The distribution is given by

$$p(x) = \begin{cases} q^{x-1}p & x = 1, 2, ... \\ 0 & \text{otherwise} \end{cases}$$

   - The mean is calculated as follows.

$$E(X) = \sum_{x=1}^{\infty} x q^{x-1} p = \sum_{x=1}^{\infty} x q^{x-1} p$$

$$E(X) = p \sum_{x=1}^{\infty} \frac{d}{dq} q^x$$

   because the sequence converges, so we can exchange the order of summation and the differentiation.

$$E(X) = p \frac{d}{dx} \sum_{x=0}^{\infty} q^x = p \frac{d}{dq} \frac{1}{1-q}$$

$$= p \frac{1}{(1-q)^2} = \frac{1}{p}$$

- Variance $V(X) = \frac{q}{p^2}$
- Example 6.11 on page 199

4. *Poisson distribution.* The Poisson distribution has the following pdf

$$p(x) = \begin{cases} \frac{e^{-\alpha}\alpha^x}{x!} & x = 0, 1, 2, ... \\ 0 & \text{otherwise} \end{cases}$$

- Poisson distribution property: mean and variance are the same

$$E(X) = \alpha = V(X)$$

- The cdf of the Poisson distribution

$$F(x) = \sum_{i=0}^{x} \frac{e^{-\alpha}\alpha^i}{i!}$$

because α is a constant and the increase rate of $i!$ is much faster than that of α^i, the distribution is mostly determined by the first a few items.

- Poisson distribution is most useful in Poisson process which is used most frequently in describing such phenomena as telephone call arrivals.
- Example 6.12, 6.13, 6.14 on page 200, page 201

5.4 Contineous Distributions

Contineous random variables can be used to describe phenomena where the values of a random variable can be any value in an interval: the time to failure, or the length of a broken rod. Seven distributions will be discussed.

1. *Uniform distributin.*

   - pdf:

$$f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

   - cdf:

$$F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & x > b \end{cases}$$

   - mean:

$$E(X) = \frac{a+b}{2}$$

- variance:
$$V(X) = \frac{(b-a)^2}{12}$$
- the interval where X can assume value can be arbitrarily long, but it cannot be infinite.
- Example 6.15 and 6.16 on page 202, 203

2. *Exponential distribution.* Exponential distributed random variable is one of most frequently used distribution in computer simulation. It is widely used in simulations of computer network and others.

   - pdf
   $$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
   - cdf
   $$F(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$
   - mean
   $$E(X) = \frac{1}{\lambda}$$
   - variance
   $$V(X) = \frac{1}{\lambda^2}$$
   - memoryless property of the exponential distributed random variables: the future values of the exponentially distributed values are not affected by the past values. Compare this to, for example, a uniformly distributed random variable, one can see the difference. For example, when throwing a fair coin, we can consider the probability of head and tail is the same which has the value of 0.5. If, after a result of head, we would *expect* to see a tail (though it may not happen). In exponentially distributed random variable, we cannot have this type of expectation. In another word, we know nothing about the future value of the random variable given a full history of the past.
   Mathematical proof.
   $$P(X > s + t | X > s) = \frac{P(X > s+t)}{P(X > s)} = \frac{e^{-\lambda(s+t)}}{e^{-\lambda s}} = e^{-\lambda t} = P(X > t)$$
   - Example 6.17 and 6.18 on page 204 and 205 where Example 6.18 demonstrates the memoryless property of the exponential distribution.

3. *Gamma distribution.*

   - pdf
   $$f(x) = \begin{cases} \frac{\beta\theta}{\Gamma(\beta)}(\beta\theta x)^{\beta-1} e^{\beta\theta x} & x > 0 \\ 0 & \text{otherwise} \end{cases}$$
   where $\Gamma(\beta) = (\beta - 1)!$ when β is an integer.

- When $\beta == 1$, this is the exponential distribution. In another word, the Gamma distribution is a more general form of exponential distribution.
- mean
$$E(X) = \frac{1}{\theta}$$
- variance
$$V(X) = \frac{1}{\beta\theta^2}$$

4. *Erlang distribution.*
   - When the parameter β in Gamma distribution is an integer, the distribution is refered to as *Erlang* distribution.
   - When $\beta = k$, a positive integer, the cdf of Erlang distribution is (using integration by parts)
   $$F(x) = \begin{cases} 1 - \sum_{i=0}^{k-1} \frac{e^{-k\theta x}(k\theta x)^i}{i!} & x > 0 \\ 0 & x \leq 0 \end{cases}$$
   which is the sum of Poisson terms with mean $\alpha = k\theta x$
   - mean
   $$E(X) = \frac{1}{\theta}$$
   - variance
   $$V(X) = \frac{1}{k\theta^2}$$
   - Example 6.19, 6.20 on page 208, 209.

5. *Normal distribution.*
   - pdf
   $$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], -\infty < x < \infty$$
   - cdf
   $$F(x) = P(X \leq x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right] dt$$
   This value is very difficult to calculate. Often a table is made for $X \sim N(0,1)$. Because an $X \sim N(\mu, \sigma^2)$ can be transformed into $X \sim N(0,1)$ by let
   $$Z = \frac{X-\mu}{\sigma}$$

- To calculate $P(X \leq x)$ for $X \sim N(\mu, \sigma^2)$, we use $\Phi(\frac{x-\mu}{\sigma})$
  Example: to calculate F(56) for N(50,9), we have

$$F(56) = \Phi(\frac{56-50}{3}) = \Phi(2) = 0.9772$$

- mean σ
- variance σ^2
- Notation: $X \sim N(\mu, \sigma^2)$
- The curve shape of the normal pdf is like a "bell".
- properties:
  - $lim_{x \to -\infty} f(x) = 0$ and $lim_{x \to \infty} f(x) = 0$
  - $f(\mu - x) = f(\mu + x)$ the pdf is symmetric about μ because of this, $\Phi(-x) = 1 - \Phi(x)$.
  - the maximum value of the pdf occurs at $x = \mu$ (thus, the mean and the mode are equal.
- Example 6.21 and 6.22 on page 211, 6.23 and 6.24 on page 213.

6. *Weibull distribution.* The random variable X has a Weibull distribution if its pdf has the form

$$f(x) = \begin{cases} \frac{\beta}{\alpha}\left(\frac{x-\gamma}{\alpha}\right)^{\beta-1} exp\left[-\left(\frac{x-\gamma}{\alpha}\right)^{\beta}\right] & x \geq \gamma \\ 0 & \text{otherwise} \end{cases}$$

- Weibull distribution has the following three parameters:
  (a) γ which has the range of $-\infty < \gamma < \infty$ which is the location parameter
  (b) α which is greater than zero which is the scale parameter
  (c) β which is a positive value determines the shape

Chapter 6

Queueing Models

A typical queueing model has a few key elements:

- customers
- servers
- network of queues

See examples listed in Table 7.1 on page 235

6.1 Characteristics

Calling population: The population of potential customers is refered to as the *calling population.* Use examples from Table 7.1

- The calling population can be finite or infinite.
- In the systems with large population, we usually assume the population is infinite.
- The key difference between "finite" and "infinite" population model is how the arrival rate is defined.
  - In an infinite population model, the arrival rate is not affected by the number of customers in the system. Usually the system is viewed as an *open* system, customers come from outside the system and leave the system after finishing the work.
  - In a finite population model, the arrival rate at a server is affected by the population in the system. Usually the system is viewed as a *closed* system, customers (with a fixed population) don't leave the system, they merely move around system from one server to another, from one queue to another.

System capacity: the maximum number of customers allowed in the system is called the *system capacity*. Examples:

- Telephone booth: capacity is one. When the server is busy (the booth occupied), the incoming customer is turned away.
- A typical TCP/IP packet buffer has a limit of 4096 bytes.
- The number of simultaneous connections allowed at a web server usually is a fixed constant.

The arrival process:

- for infinite population
  - inter-arrival time characterized by a random distribution, e.g. Poisson distribution
  - scheduled arrivals, such as patients to a physician's office; airline flight arrives at an airport. The interarrival time may be a constant, or a constant with a small random fluctuation.
  - at least one customer is assumed to always be present in the queue, so the server is never idle because of the lack of the customers, e.g. parts in an assembly line.
- for finite population:
  - Define a customer is *pending* when it is outside queueing system, and it is a member of the potential calling population.
    E.g. when simulating a local area network, if a particular computer is powered off, we say it is *pending*. As soon as it is powered on, the customer (computer) will demand service from the network.
  - Define *runtime* of a given customer as the length of the time from departure from the queueing system until the customer's next arrival to the queue. Runtime essentially is the time when the customer is being serviced.

  The arrival rate in a finite population model is a function of the number of pending customers.

6.2 Queue Bahavior and Queue Discipline

Queue behavior: how customers react to the situation of the queue.

balk: Customers leave when they see the queue is too long.

renege: Customers leave after being in the queue when they see the queue is moving too slowly.

jockey: Customers move from one queue to another if they think they have chosen a slow line.

Queue discipline: how customers are served in the queue.

FIFO: first-in-first-out

LIFO: last-in-first-out

SIRO: service in random order

SPT: shortest processing time first

PR: service according to priority

RR: round robin

6.3 Service Time and Service Mechanisms

- Service time of a successive arrivals are denoted by $S_1, S_2, S_3, ...$ They may be constants or random variables. If they are random variables, they follow certain distribution.
- A queueing system consists of service centers and inter-connecting queues.
- Each service cetner contains a number of servers, c.
  - $c = 1$ single server
  - $1 < c < \infty$ multiple servers
  - $c = \infty$ unlimited servers
- Transition state and steady state: simulation needs some time to "warm up" from a initial state to a steady state. The status in a transition state should not be considered as indicative as typical measures. One should always wait until certain time elapses before collecting statistics.
- Examples 7.1 and 7.2 on page 239

6.4 Queueing Notation

A queueing system is often noted by A/B/c/N/K where

A: interarrival time distribution

B: service time distribution

c: number of parallel servers

N: queueing capacity

K: size of the calling population

A few notes:

- A few special notations: M for exponential distribution, G for general distribution, D for constant.
- When the queueing capacity or the calling population size, or both are infinite, they can be omitted from the notation, e.g. M/M/1 meaning expoential distribution for arrival and service time with one server, the queueing capacity and the calling population size are both infinite.

See Example 7.3 on page 241-242

6.5 Long Term Measures of Performance of Queueing Systems

- time-averaged number of customers in the system (L) and in the queue (L_Q)
- average time in system (W)
- average time in queue (W_Q)
- Little's Law (consevation equation) in a queueing system, if the queue length is denoted by L, the arrival rate is denoted by lambda, and the waiting time is denoted by W then

$$L = \lambda W$$

  intuitively, it says the queue length is proportional to the arrival rate and the waiting time. Waiting time is affected by two parameters the arrival rate and the service rate.

6.6 Markov Models and Its Evaluation

In a queueing system, if the service time and inter-arrival time are both expoentially distributed, denoted by μ and λ respectively plus

- the system is in steady state
- the population is infinite

then we can evaluate the queueing system using the so called *Markov Model*.

Let P_i represent the probability the system has i customers (including the ones in queue and ones in server).

Using the principle that the flow into a state is balanced by the flow out of the state, we have

$$
\begin{aligned}
P_0\lambda &= P_1\mu \\
P_1\lambda &= P_2\mu \\
&\ldots
\end{aligned}
$$

solve this system of equations we get

$$P_i = P_0(\frac{\lambda}{\mu})^i \quad \text{for } i > 0$$

Using the relation

$$\sum_{i=0}^{\infty} P_i = 1$$

and let

$$\frac{\lambda}{\mu} = \rho$$

we have

$$\sum_{i=0}^{\infty} P_0\rho^i = 1$$

thus

$$P_0 = \frac{1}{\sum_{i=0}^{\infty} \rho^i}$$

where

$$\sum_{i=0}^{\infty} \rho^i = \frac{1}{1-\rho}$$

so

$$P_0 = 1 - \rho$$

The meaning of the probability with no customer in the system is the same as the server is idle. So the measure ρ can be considered as the probability that the server is busy, which is the utilization of the server. With P_0 solved, all other items can be solved.

$$P_i = (1-\rho)\rho^i$$

Use P_is, we can obtain all other measures of interest.

Average number of customers in system :

$$L = \sum_{i=0}^{\infty} i * P_i = \sum_{i=0}^{\infty} i * (1-\rho) * \rho^i = \frac{\rho}{1-\rho}$$

waiting time in system: use Little's Law

$$W = \frac{L}{\lambda} = \frac{1}{\mu(1-\rho)}$$

waiting time in the queue: this can be calculated using the fact that waiting time in the queue is the total waiting time less the time spent in the server. The time spent in the server is the average service time $\frac{1}{\mu}$ (note that μ is the service rate here) so

$$W_Q = W - \frac{1}{\mu} = \frac{\rho}{\mu(1-\rho)}$$

Average queue length: use the Little's Law again

$$L_Q = \lambda W_Q = \frac{\rho^2}{1-\rho}$$

Chapter 7

Random-Number Generation

Random numbers play a key role in discrete event simulation. We have used the uniformly distributed random numbers in many programming assignments before simulation. In simulation, we need random numbers with other distribution besides the uniformly distributed one.

7.1 Properties of Random Numbers

- A sequence of random numbers, $R_1, R_2, R_3, ...$ must have two important properties:
  1. uniformity, i.e. they are equally probable every where
  2. independence, i.e. the current value of a random variable has no relation with the previous values
- Each random number R_i is an independent sample drawn from a continueous uniform distribution between zero and one.
  - pdf
  $$f(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$
  - expectation
  $$E(R) = \int_0^1 x dx = \frac{1}{2}$$
  - variance
  $$V(R) = \int_0^1 x^2 dx - [E(R)]^2 = \frac{1}{12}$$
- Some consequences of the uniformity and independence properties

1. If the interval (0,1) is divided into n sub-intervals of equal length, the expected number of observations in each interval is N/n where N is the total number of observations. Note that N has to be sufficiently large to show this trend.
2. The probability of observing a value in a particular interval is independent of the previous values drawn.

7.2 Generation of Pseudo-Random Numbers

- In computer simulation, we often do not want to have pure random numbers because we would like to have the control of the random numbers so that the experiment can be repeated.
- In general, a systematic way to generate *pseudo-random* number is used to generate the random numbers used in simulation. Some *algorithms* are needed.
- We generate the uniformly distributed random numbers first; then we use this to generate random numbers of other distribution.
- Some desired properties of pseudo-random number generators:
  - The routine should be fast.
  - The routine should be portable across hardware platforms and programming languages.
  - The routine should have sufficiently long cycle.
    * A *cycle length* represents the length of the random number sequence before previous numbers begin to repeat themselves in an earlier order. For example
      `4,9,5,6,9,3,8, 4,9,5,6,9,3,8, 4,9,5,6,9,3,8, ...`
      appears to have a cycle length of 7 (this is just an example of cycle, a random number of cycle 7 is completely unaccetable!).
    * A special case of cycling is *degenerating* where the same random numbers appear repeatedly.
    * Because we use an algorithm to generate random number, cycling cannot be avoided. But long cycles (e.g. a few millions or a few billions) serve the purpose of general simulations.
  - The random numbers should be replicable.
  - Most importantly, the generated random numbers should closely approximate the ideal statistical properties of uniformity and independence.

7.3 Techniques for Generating Random Numbers

Many different methods of generating pseudo-random numbers are available. This text introduces two of them, with one in great detail.

7.3.1 Linear Congruential Method

- The linear congruential method produces a sequence of integers $X_1, X_2, X_3, ...$ between zero and *m-1* according to the following recursive relationship:

$$X_{i+1} = (aX_i + c) \bmod m, \qquad i = 0, 1, 2, ...$$

  - The initial value X_0 is called the seed;
  - a is called the constant multiplier;
  - c is the increment
  - m is the modulus

  The selection of *a, c, m* and X_0 drastically affects the statistical properties such as mean and variance, and the cycle length.

- When $c \neq 0$, the form is called the *mixed congruential method*; When *c = 0*, the form is known as the *multiplicative congruential method*.
- Example 8.1 on page 292
- Issues to consider:
  - The numbers generated from the example can only assume values from the set *I* = {*0, 1/m, 2/m, ..., (m-1)/m*}. If m is very large, it is of less problem. Values of $m = 2^{31} - 1$ and $m = 2^{48}$ are in common use.
  - To achieve maximum density for a given range, proper choice of *a, c, m* and X_0 is very important. Maximal period can be achieved by some proven selection of these values.
    * For m a power of 2, i.e. $m = 2^b$, and $c \neq 0$, the longest possible period is $P = m = 2^b$, when c is relatively prime to m and *a = 1 + 4 k* where k is an integer.
    * For m a power of 2, i.e. $m = 2^b$, and $c = 0$, the longest possible period is $P = m/4 = 2^{b-2}$, when X_0 is odd and the multiplier, a is given by $a = 3 + 8k$ or $a = 5 + 8k$ where k is an integer.
    * For m a prime number and *c = 0*, the longest possible period is *P = m - 1* when a satisfies the property that the smallest k such that $a^k - 1$ is divisible by m is *k = m - 1*.

      For example, we choose *m = 7* and *a = 3*, the above conditions satisfy. Here k has to be 6.
      · when k = 6, $a^k - 1 = 728$ which is divisible by m
      · when k = 5, $a^k - 1 = 242$ which is not divisible by m
      · when k = 4, $a^k - 1 = 80$ which is not divisible by m
      · when k = 3, $a^k - 1 = 26$ which is not divisible by m

      Of course, the longest possible period here is 6, which is of no practical use. But the example shows how the conditions can be checked.

- Examples 8.2, 8.3 and 8.4 on page 294 and page 295.

7.3.2 Combined Linear Congruential Generators

By combining two or more multiplicative congruential generators may increase the length of the period and results in other better statistics. See Example 8.5 on page 297.

7.4 Tests for Random Numbers

When a random number generator is devised, one needs to test its property. The two properties we are concerned most are uniformity and independence. A list of tests will be discussed. The first one tests for uniformity and the second to fifth ones test independence.

1. Frequency test
2. Runs test
3. Autocorrelation test
4. Gap test
5. Poker test

The algorithms of testing a random number generator are based on some statistics theory, i.e. testing the hypotheses. The basic ideas are the following, using testing of uniformity as an example.

We have two hypotheses, one says the random number generator is indeed uniformly distributed. We call this H_0, known in statistics as *null hypothesis*. The other hypothesis says the random number generator is not uniformly distributed. We call this H_1, known in statistics as *alternative hypothesis*.

We are interested in testing result of H_0, reject it, or fail to reject it.

To see why we don't say *accept H null*, let's ask this question: what does it mean if we had said *accepting H null*? That would have meant the distribution is truely uniform. But this is impossible to state, without exhaustive test of a *real* random generator with infinite number of cases. So we can only say *failure to reject H null*, which means no evidence of non-uniformity has been detected on the basis of the test. This can be described by the saying "so far so good".

On the other hand, if we have found evidence that the random number generator is not uniform, we can simply say *reject H null*.

It is always possible that the H_0 is true, but we rejected it because a sample landed in the H_1 region, leading us to reject H_0. This is known as *Type I* error. Similarly if H_0 is false, but we didn't reject it, this also results in an error, known as *Type II* error.

With these information, how do we state the result of a test? (How to perform the test will be the subject of next a few sections)

- A level of statistical significance α has to be given. The level α is the probability of rejecting the H null while the H null is true (thus, Type I error).

$$\alpha = P(\text{reject } H_0 | H_0 \text{ true})$$

- We want the probability as little as possible. Typical values are 0.01 (one percent) or 0.05 (five percent).
- Decreasing the probability of Type I error will increase the probability of Type II error. We should try to strike a balance.

For a given set of random numbers produced by a random number generator, the more tests are, the more accurate the results will be.

7.4.1 Frequency test

- The frequency test is a test of uniformity.
- Two different methods available, Kolmogorov-Smirnov test and the chi-square test. Both tests measure the agreement between the distribution of a sample of generated random numbers and the theoretical uniform distribution.
- Both tests are based on the null hypothesis of no significant difference between the sample distribution and the theoretical distribution.

The Kolmogorov-Smirnov test This test compares the cdf of uniform distribution *F(x)* to the empirical cdf $S_N(X)$ of the sample of N observations.

- $F(x) = x \quad 0 \leq x \leq 1$
- $S_N(x) = \frac{\text{number of } R_1, R_2, \ldots, R_N \leq x}{N}$
- As N becomes larger, $S_N(x)$ should be close to *F(x)*
- Kolmogorov-Smirnov test is based on the statistic

$$D = \max|F(x) - S_N(x)|$$

  that is the absolute value of the differences.
- Here D is a random variable, its sampling distribution is tabulated in Table A.8.
- If the calcualted D value is greater than the ones listed in the Table, the hypothesis (no disagreement between the samples and the theoretical value) should be rejected; otherwise, we don't have enough information to reject it.

- Following steps are taken to perform the test.
  1. Rank the data from smallest to largest
  $$R_{(1)}, \leq R_{(2)}, \leq ... \leq R_{(N)}$$
  2. Compute
  $$D^+ = \max_{1\leq i\leq N}\left\{\frac{i}{N} - R_{(i)}\right\}$$
  $$D^- = \max_{1\leq i\leq N}\left\{R_{(i)} - \frac{i-1}{N}\right\}$$
  3. Compute $D = \max(D^+, D^-)$
  4. Determine the critical value, D_α, from Table A.8 for the specified significance level α and the given sample size N.
  5. If the sample statistic D is greater than the critical value D_α, the null hypothsis that the sample data is from a uniform distribution is rejected; if $D \leq D_\alpha$, then there is no evidence to reject it.
- Example 8.6 on page 300.

Chi-Square test The chi-square test looks at the issue from the same angle but uses different method. Instead of measure the difference of each point between the samples and the true distribution, chi-square checks the "deviation" from the "expected" value.

$$x_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where n is the number of classes (e.g. intervals), O_i is the number of samples obseved in the interval, E_i is expected number of samples in the interval. If the sample size is N, in a uniform distribution,

$$E_i = \frac{N}{n}$$

See Example 8.7 on page 302.

7.4.2 Runs Tests

1. Runs up and down

- The runs test examines the arrangement of numbers in a sequence to test the hypothesis of independence.
- See the tables on page 303. With a closer look, the numbers in the first table go from small to large with certain pattern. Though these numbers will pass the uniform tests in previous section, they are not quite independent.

- A run is defined as a succession of similar events proceded and followed by a different event. E.g. in a sequence of tosses of a coin, we may have

  ```
  H T T H H T T T H T
  ```

  The first toss is proceded and the last toss is followed by a "no event". This sequence has six runs, first with a length of one, second and third with length two, fourth length three, fifth and sixth length one.

- A few features of a run
  - two characteristics: number of runs and the length of run
  - an up run is a sequence of numbers each of which is succeeded by a larger number; a down run is a squence of numbers each of which is succeeded by a smaller number

- If a sequence of numbers have too few runs, it is unlikely a real random sequence. E.g. 0.08, 0.18, 0.23, 0.36, 0.42, 0.55, 0.63, 0.72, 0.89, 0.91, the sequence has one run, an up run. It is not likely a random sequence.

- If a sequence of numbers have too many runs, it is unlikely a real random sequence. E.g. 0.08, 0.93, 0.15, 0.96, 0.26, 0.84, 0.28, 0.79, 0.36, 0.57. It has nine runs, five up and four down. It is not likely a random sequence.

- If a is the total number of runs in a truly random sequence, the mean and variance of a is given by

  $$\mu_a = \frac{2N-1}{3}$$

  and

  $$\sigma^2 = \frac{16N-29}{90}$$

- For $N > 20$, the distribution of a is reasonably approximated by a normal distribution, $N(\mu_a, \sigma_a^2$. Converting it to a standardized normal distribution by

  $$Z_0 = \frac{a-\mu_a}{\sigma_a}$$

  that is

  $$Z_0 = \frac{a-[(2N-1)/3]}{\sqrt{(16N-29)/90}}$$

- Failure to reject the hypothesis of independence occurs when $-z_{\alpha/2} \le Z_0 \le z_{\alpha/2}$, where the α is the level of significance.

- See Figure 8.3 on page 305

- See Example 8.8 on page 305

2. Runs above and below the mean.

- The previous test for up runs and down runs are important. But they are not adquate to assure that the sequence is random.
- Check the sequence of numbers at the top of page 306, where they pass the runs up and down test. But it display the phenomenon that the first 20 numbers are above the mean, while the last 20 are below the mean.
- Let n_1 and n_2 be the number of individual observations above and below the mean, let b the total number of runs.
- For a given n_1 and n_2, the mean and variance of b can be expressed as

$$\mu_b = \frac{2n_1n_2}{N} + \frac{1}{2}$$

  and

$$\sigma_b^2 = \frac{2n_1n_2(2n_1n_2 - N)}{N^2(N-1)}$$

- For either n_1 or n_2 greater than 20, b is approximately normally distributed

$$Z_0 = \frac{b - (2n_1n_2/N) - 1/2}{[\frac{2n_1n_2(2n_1n_2-N)}{N^2(N-1)}]^{1/2}}$$

- Failure to reject the hypothesis of independence occurs when $-z_{\alpha/2} \leq Z_0 \leq z_{\alpha/2}$, where α is the level of significance.
- See Example 8.9 on page 307

3. Runs test: length of runs.

- The example in the book:

  `0.16, 0.27, 0.58, 0.63, 0.45, 0.21, 0.72, 0.87, 0.27, 0.15, 0.92, 0.85,...`

  If the same pattern continues, two numbers below average, two numbers above average, it is unlikely a random number sequence. But this sequence will pass other tests.
- We need to test the randomness of the length of runs.
- Let Y_i be the number of runs of length i in a sequence of N numbers. E.g. if the above sequence stopped at 12 numbers (N = 12), then $Y_1 = Y_3 = Y_4 = ... = Y_{11} = 0$ and $Y_2 = 6$

Obviously Y_i is a random variable. Among various runs, the expected value for runs up and down is given by

$$E(Y_i) = \frac{2}{(i+3)!}[N(i^2+3i+1) - (i^3+3i^2-i-4)] \quad \text{for} \quad i \leq N-2$$

and

$$E(Y_i) = \frac{2}{N!} \quad \text{for} \quad i = N-1$$

- The number of runs above and below the mean, also random variables, the expected value of Y_i is approximated by

$$E(Y_i) = \frac{Nw_i}{E(I)}, N > 20$$

  where *E(I)* the approximate expected length of a run and w_i is the approximate probability of length i.

- w_i is given by

$$w_i = \left(\frac{n1}{N}\right)^i \left(\frac{n2}{N}\right) + \left(\frac{n1}{N}\right) \left(\frac{n2}{N}\right)^i$$

- *E(I)* is given by

$$E(I) = \frac{n_1}{n_2} + \frac{n_2}{n_1} \quad N > 20$$

- The approximate expected total number of runs (of all length) in a sequence of length N is given by

$$E(A) = \frac{N}{E(I)}, N > 20$$

  (total number divided by expected run length).

- The appropriate test is the chi-square test with O_i being the observed number of runs of length i

$$x_0^2 = \sum_{i=1}^{L} \frac{[O_i - E(Y_i)]^2}{E(Y_i)}$$

  where L = *N - 1* for runs up and down, $L = N$ for runs above and belown the mean.

- See Example 8.10 on page 308 for length of runs up and down.

- See Example 8.11 on page 311 for length of above and below the mean.

7.4.3 Tests for Auto-correlation

- The tests for auto-correlation are concerned with the dependence between numbers in a sequence.
- The list of the 30 numbers on page 311 appears to have the effect that every 5th number has a very large value. If this is a regular pattern, we can't really say the sequence is random.
- The test computes the auto-correlation between every m numbers (m is also known as the lag) starting with the ith number.

  Thus the autocorrelation ρ_{im} between the following numbers would be of interest.

  $$R_i, R_{i+m}, R_{i+2m}, ...R_{i+(M+1)m}$$

  The value M is the largest integer such that $i + (M+1)m \leq N$ where N is the total number of values in the sequence.

  E.g. N = 17, i = 3, m = 4, then the above sequence would be 3, 7, 11, 15 (M = 2). The reason we require *M+1* instead of *M* is that we need to have at least two numbers to test (M = 0) the autocorrelation.
- Since a non-zero autocorrelation implies a lack of independence, the following test is appropriate

  $H_0: \quad \rho_{im} = 0$
  $H_1: \quad \rho_{im} \neq 0$
- For large values of *M*, the distribution of the estimator ρ_{im}, denoted as $\hat{\rho_{im}}$, is approximately normal if the values $R_i, R_{i+m}, R_{i+2m}, ...R_{i+(M+1)m}$ are uncorrelated.
- Form the test statistic

  $$Z_0 = \frac{\hat{\rho_{im}}}{\sigma_{\hat{\rho_{im}}}}$$

  which is distributed normally with a mean of zero and a variance of one.
- The actual formula for $\hat{\rho_{im}}$ and the standard deviation is

  $$\hat{\rho_{im}} = \frac{1}{M+1}\left[\sum_{k=0}^{M} R_{i+km}R_{(k+1)m}\right] - 0.25$$

  and

  $$\sigma_{\hat{\rho_{im}}} = \frac{\sqrt{13M+7}}{12(M+1)}$$
- After computing Z_0, do not reject the null hypothesis of independence if

  $$-z_{\alpha/2} \leq Z_0 \leq z_{\alpha/2}$$

  where α is the level of significance.
- See Example 8.12 on page 312.

7.4.4 Gap Test

- The gap test is used to determine the significance of the interval between recurrence of the same digit.
- A gap of length x occurs between the recurrence of some digit.
- See the example on page 313 where the digit 3 is underlined. There are a total of eighteen 3's in the list. Thus only 17 gaps can occur.
- The probability of a particular gap length can be determined by a Bernoulli trail.

  $$P(\text{gap of } n) = P(x \neq 3)P(x \neq 3)...P(x \neq 3)P(x = 3)$$

  If we are only concerned with digits between 0 and 9, then

  $$P(\text{gap of } n) = 0.9^n 0.1$$

- The theoretical frequency distribution for randomly ordered digits is given by

  $$P(gap \leq x) = F(x) = 0.1 \sum_{n=0}^{x} (0.9)^n = 1 - 0.9^{x+1}$$

- Steps involved in the test.

  Step 1. Specify the cdf for the theoretical frequency distribution given by Equation (8.14) based on the selected class interval width (See Table 8.6 for an example).

  Step 2. Arrange the observed sample of gaps in a cumulative distribution with these same classes.

  Step 3. Find D, the maximum deviation between *F(x)* and $S_N(x)$ as in Equation 8.3 (on page 299).

  Step 4. Determine the critical value, D_α, from Table A.8 for the specified value of α and the sample size N.

  Step 5. If the calculated value of D is greater than the tabulated value of D_α, the null hypothesis of independence is rejected.

- See the Example 8.13 on page 314

7.4.5 Poker Test

- The poker test for independence is based on the frequency in which certain digits are repeated in a series of numbers.
- For example 0.255, 0.577, 0.331, 0.414, 0.828, 0.909, 0.303, 0.001... In each case, a pair of like digits appears in the number.

- In a three digit number, there are only three possibilities.
  1. The individual digits can be all different. Case 1.
  2. The individual digits can all be the same. Case 2.
  3. There can be one pair of like digits. Case 3.
- P(case 1) = P(second differ from the first) * P(third differ from the first and second) = 0.9 * 0.8 = 0.72

  P(case 2) = P(second the same as the first) * P(third same as the first) = 0.1 * 0.1 = 0.01
  P(case 3) = 1 - 0.72 - 0.01 = 0.27
- See Example 8.14 on page 316

Chapter 8

Random Variate Generation

Now that we have learned how to generate a uniformly distributed random variable, we will study how to produce random variables of other distribution using the uniformly distributed random variable.

The techniques discussed include inverse transform and convolution. Also discussed is the acceptance-rejection technique.

All work here assume the existence of a source of uniform (0,1) random numbers, $R_1, R_2, ...$

- pdf

$$f_R(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

- cdf

$$F_R(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x \leq 1 \\ 1, & x > 1 \end{cases}$$

8.1 Inverse Transform Technique

- The inverse transform technique can be used to sample from exponential, the uniform, the Weibull and the triangle distributions.
- The basic principle is to find the inverse function of F, F^{-1} such that $F\ F^{-1} = F^{-1}\ F = I$.
- F^{-1} denotes the solution of the equation $r = F(x)$ in terms of r, not $1/F$. For example, the inverse of y = x is x = y, the inverse of y = 2 x + 1 is x = (y-1)/2 ...

8.1.1 Exponential Distribution

- pdf

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- cdf

$$F(x) = \int_{-\infty}^{x} f(t)dt = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- The idea is to solve $y = 1 - e^{-\lambda x}$ for x where y is uniformly distributed on (0,1) because it is a cdf. Then x is exponentially distributed.
- This method can be used for any distribution in theory. But it is particularly useful for random variates that their inverse function can be easily solved.
- Steps involved are as follows.

Step 1. Compute the cdf of the desired random variable X.
For the exponential distribution, the cdf is $F(x) = 1 - e^{-\lambda x}$.

Step 2. Set $R = F(X)$ on the range of X.
For the exponential distribution, $R = 1 - e^{-\lambda x}$ on the range of $x \geq 0$.

Step 3. Solve the equation $F(X) = R$ for X in terms of R.
For the exponential distribution, the solution proceeds as follows.

$$\begin{array}{ll} 1 - e^{-\lambda X} & = R \\ e^{-\lambda X} & = 1 - R \\ -\lambda X & = ln(1 - R) \\ X & = \frac{-1}{\lambda} ln(1 - R) \end{array}$$

Step 4. Generate (as needed) uniform random numbers $R_1, R_2, ...$ and compute the desired random variates by

$$X_i = F^{-1}(R_i)$$

In the case of exponential distribution

$$X_i = \frac{-1}{\lambda} ln(1 - R_i)$$

for i = *1, 2, 3, ...* where R_i is a uniformly distributed random number on (0,1).
In practice, since both R_i AND $1 - R_i$ are uniformly distributed random number, so the calculation can be simplified as

$$X_i = \frac{-1}{\lambda} ln\ R_i$$

- To see why this is correct, recall
$$X_i = F^{-1}(R_i)(*)$$
and
$$R_i = F(X_i)(**)$$
Because $X_i \le x_0$ is equivelant to $F^{-1}(R_i) \le x_0$, and F is a non-decreasing function (so that if $x \le y$ then $F(x) \le F(y)$) we get $X_i \le x_0$ is equivelant to $F^{-1}(R_i) \le x_0$, which implies that $F(F^{-1}(R_i)) \le F(x_0)$ which is equivelant to $R_i \le F(x_0)$.

  This means
$$P(X_i \le x_0) = P(R_i \le F(x_0))$$
Since R_i is uniformly distributed on (0,1) and $F(x_0)$ is the cdf of exponential function which is between 0 and 1, so
$$P(R_i \le F(x_0)) = F(x_0)$$
which means
$$P(X_i \le x_0) = F(x_0)$$
This says the $F(x_0)$ is the cdf for X and X has the desired distribution.

Once we have this procedure established, we can proceed to solve other similar distribution for which a inverse function is relatively easy to obtain and has a closed formula.

8.1.2 Uniform Distribution

- If we want a random variate X uniformly distributed on the interval [a,b], a reasonable guess for generating X is given by
$$X = a + (b-a)R$$
where R is uniformly distributed on (0,1).

  If we follow the steps outlined in previous section, we get the same result.

- pdf for X
$$f(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & otherwise \end{cases}$$

- steps:

  Step 1. the cdf
$$F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x \le b \\ 1 & x > b \end{cases}$$

  Step 2. Set $F(X) = (X-a)/(b-a) = R$

Step 3. Solve for X is terms of R yields

$$X = a + (b - a)R$$

which is the same as the earlier guess.

Step 4. Generate R_i as needed, calcualte X_i using the function obtained.

8.1.3 Weibull Distribution

- pdf

$$f(x) = \begin{cases} \frac{\beta}{\alpha^\beta} x^{\beta-1} e^{-(x/\alpha)^\beta}, & x \geq 0 \\ 0, & otherwise \end{cases}$$

- Steps:

Step 1. cdf

$$F(x) = 1 - e^{-(x/\alpha)^\beta}$$

for $x \geq 0$

Step 2. let $F(x) = 1 - e^{-(x/\alpha)^\beta} = R$

Step 3. Solve for X in terms of R yields

$$X = \alpha[-ln(1 - R)]^{1/\beta}$$

Step 4. Generate uniformly distributed R_i from (0,1) feeding them to the function in Step 3 to get X.

8.1.4 Triangular Distribution

- cdf

$$f(x) = \begin{cases} x & 0 \leq x \leq 1 \\ 2 - x & 1 < x \leq 2 \\ 0 & otherwise \end{cases}$$

Its curve is shown on page 328

- Steps

Step 1. cdf

$$F(x) = \begin{cases} 0 & x \leq 0 \\ \frac{x^2}{2} & 0 < x \leq 1 \\ 1 - \frac{(2-x)^2}{2} & 1 < x \leq 2 \\ 1 & x > 2 \end{cases}$$

Step 2. let $R = \frac{X^2}{2}$ for $0 < X \leq 1$ and $R = 1 - \frac{(2-x)^2}{2}$ for $1 < x \leq 2$
Step 3. Solve X in terms of R

$$X = \begin{cases} \sqrt{2R} & 0 \leq R \leq \frac{1}{2} \\ 2 - \sqrt{2(1-R)} & \frac{1}{2} < R \leq 1 \end{cases}$$

8.1.5 Empirical Continuous Distributions

- If the modeler has been unable to find a theoretical distribution that provides a good model for the input data, it may be necessary to use the empirical distribution of the data.
- A typical way of resolving this difficult is through "curve fitting".
- Steps involved: (see Exmaple 9.2 on page 328)
  - Collect empirical data and group them accordingly.
  - Tabulate the frequency and cumulative frequency.
  - Now assume the value of cumulative frequency as a function of the empirical data, i.e. F(x) = r
  - Establish a relation between x and r using linear interpolation

    $$X_1 = x_a + \frac{R_1 - y_a}{y_b - y_a}(x_b - x_a)$$

    for each of the intervals.
- Example 9.3 on page 332: note that in this example, five response time are used which are all distinct. If there are values are the same, or if the number of samples are large, we can certainly group them in the different ways.

8.1.6 Continuous Distributions without a Closed-Form Inverse

- Many continuous distributions don't have a closed-form inverse function. Strictly speaking, the inverse transform method discussed earlier cannot be applied.
- If we are willing to accept numeric solution, inverse functions can be found. Here we have an example for normal distribution.
- One of the inverse cdf of the standard normal distribution was proposed by Schmeiser:

  $$X = F^{-1}(R) \approx \frac{R^{0.135} - (1-R)^{0.135}}{0.1975}$$

  for $0.0013499 \leq R \leq 0.9986501$ which matches the true normal distribution with one digit after decimal point. See Table 9.6 on page 335.

8.1.7 Discrete Distribution

- All discrete distributions can be generated using the inverse transform technique.
- This section discusses the case of empirical distribution, (discrete) uniform distribution, and geometric distribution.
- Empirical discrete distribution. The idea is to collect and group the data, then develop the pdf and cdf. Use this information to obtain F^{-1} so that $X = F^{-1}(R)$ will be the random number function that we look for.

  Example 9.4 on page 336.
- Discrete Uniform Distribution (Example 9.5 on page 338)
  - pdf
  $$p(x) = \frac{1}{k} \quad x = 1, 2, ...k$$
  - cdf
  $$F(x) = \begin{cases} 0 & x < 1 \\ \frac{i}{k} & i \le x < k \text{ for } 1 \le i < k \\ 1 & k \le x \end{cases}$$
  - Let *F(X) = R*
  - Solve X in terms of R. Since x is discrete,
  $$r_{i-1} = \frac{i-1}{k} < R \le r_i = \frac{i}{k}$$
  thus,
  $$i - 1 < Rk \le i$$
  $$Rk \le i < Rk + 1$$
  Consider the fact that i and k are integers and R is between (0,1). For the above relation to hold, we need
  $$X = \lceil Rk \rceil$$
  - For example, to generate a random variate X, uniformly distributed on {1, 2, ..., 10 }

    (thus k = *10*)
  $$\begin{array}{ll} R_1 = 0.78 & X_1 = \lceil 0.78 * 10 \rceil = 8 \\ R_2 = 0.03 & X_2 = \lceil 0.03 * 10 \rceil = 1 \\ R_3 = 0.23 & X_3 = \lceil 0.23 * 10 \rceil = 3 \\ R_4 = 0.97 & X_4 = \lceil 0.97 * 10 \rceil = 10 \end{array}$$
- Example 9.6 on page 339 gives us another flavor. When an inverse function F^{-1} has more than one solution, one has to choose which one to use. In the example, one results in positive value and the other results in negative value. The choice is obvious.
- Example 9.7 on page 340: Geometric distribution.

- pmf

$$p(x) = p * (1-p)^x, \quad x = 0, 1, 2, ...$$

where $0 < p < 1$

- cdf

$$\begin{aligned} F(x) &= \textstyle\sum_{j=0}^{x} p(1-p)^j \\ &= \frac{p\{1-(1-p)^{x+1}\}}{1-(1-p)} \\ &= 1-(1-p)^{x+1} \end{aligned}$$

- Let R = F(x), solve for x in term of R. Because this is a discrete random variate, use the inequality (9.12) on page 337,

$$F(x_{i-1}) = r_{i-1} < R \le r_i = F(x_i)$$

that is

$$F(x_{i-1}) = r_{i-1} = 1-(1-p)^x < R \le 1-(1-p)^{x+1} = r_i = F(x_i)$$

$$(1-p)^{x+1} \le 1-R < (1-p)^x$$

$$(x+1)ln(1-p) \le ln(1-R) < xln(1-p)$$

Notice that $ln(1-p) < 0$

$$\frac{ln(1-R)}{ln(1-p)} - 1 \le x < \frac{ln(1-R)}{ln(1-p)}$$

Consider that x must be an integer, so

$$X = \lceil \frac{ln(1-R)}{ln(1-p)} - 1 \rceil$$

- Let $\beta = -1/ln(1-p)$ the equation above becomes

$$X = \lceil -\beta ln(1-R) - 1 \rceil$$

The item in the ceiling function before subtracting one is the function to generate exponentially distributed variate.

- Thus one way to generate geometric distribution is to
  1. let $\lambda = \beta^{-1} = -ln(1-p)$ as the parameter to the exponential distribution,
  2. generate an exponentially distributed variate by

  $$-\frac{1}{\lambda} ln(1-R)$$

  3. subtract one and take the ceiling
- Example 9.8 on page 341

8.1.8 Direct Transformation for the Normal Distribution

In the previous section, a simple, but less accurate method of generating a normal distribution was presented. Here we consider a direct transormation.

- Let Z_1, Z_2 be two standard normal random variates.
- Plot the two as a point in the plane and represent them in a polar coordinate system as

$$Z_1 = B\cos\theta$$
$$Z_2 = B\sin\theta$$

- It is known that $B^2 = {Z_1}^2 + {Z_2}^2$ has the chi-square distribution with 2 degrees of freedom, which is equivalent to an exponential distribution with mean 2

$$Y = \lambda e^{-\lambda t}, \quad t \geq 0$$

$$E[Y] = 2 = \lambda$$

  thus the raidus B can be generated using (9.3)

$$B = \sqrt{-2\ln R}$$

- So a normal distribution can be generated by any one of the following.

$$Z_1 = \sqrt{-2\ln R_1}\cos(2\pi R_2)$$

$$Z_2 = \sqrt{-2\ln R_1}\sin(2\pi R_2)$$

  where R_1 and R_2 are uniformly distributed over (0,1).
- To obtain normail variates X_i with mean μ and variance σ^2, transform

$$X_i = \mu + \sigma Z_i$$

8.1.9 Convolution Method

- The probability distribution of a sum of two or more independent random variables is called a *convolution* of the distributions of the original variables.
- Erlang distribution.
  - an Erlang random variable X with parameters (K, θ) can be shown to be the sum of K independent exponential random variables X_i $(i = 1, ..., K)$, each having a mean $1/K\theta$

$$X = \sum_{i=1}^{K} X_i$$

- Using equation (9.3) that can generate exponential variable, an Erlang variate can be generated by

$$\begin{aligned} X &= \sum_{i=1}^{K} \frac{-1}{K\theta} \ln R_i \\ &= \frac{-1}{K\theta} \ln \left(\prod_{i=1}^{K} R_i \right) \end{aligned}$$

- Example 9.9 on page 343

8.2 Acceptance-Rejection Technique to Generate Random Variate

- Example: use following steps to generate uniformly distributed random numbers between 1/4 and 1.

  Step 1. Generate a random number R

  Step 2a. If $R \geq 1/4$, accept $X = R$, goto Step 3

  Step 2b. If $R < 1/4$, reject R, return to Step 1

  Step 3. If another uniform random variate on [1/4, 1] is needed, repeat the procedure begining at Step 1. Otherwise stop.

  - Do we know if the random variate generated using above methods is indeed uniformly distributed over [1/4, 1]? The answer is Yes. To prove this, use the definition. Take any $1/4 \leq a < b \leq 1$,

    $$P(a < R \leq b | 1/4 \leq R \leq 1) = \frac{P(a < R \leq b)}{P(1/4 \leq R \leq 1)} = \frac{b-a}{3/4}$$

    which is the correct probability for a uniform distribution on [1/4,1].
  - The efficiency: use this method in this particular example, the rejection probability is 1/4 on the average for each number generated. The number of rejections is a geometrically distributed random variable with probability of "success" being $p = 3/4$, mean number of rejections is *(1/p - 1) = 4/3 - 1 = 1/3* (i.e. 1/3 waste).
  - For this reason, the inverse transform (X = 1/4 + (3/4) R) is more efficient method.

- **Poisson Distribution**
  - pmf

    $$p(n) = P(N = n) = \frac{e^{-\alpha}\alpha^n}{n!}, n = 0, 1, 2, ...$$

    where N can be interpreted as the number of arrivals in one unit time.
  - From the original Poisson process definition, we know the interarrival time $A_1, A_2, ...$ are exponentially distributed with a mean of α, i.e. α arrivals in one unit time.

- Relation between the two distribution:

$$N = n$$

if and only if

$$A_1 + A_2 + ... + A_n \leq 1 < A_1 + A_2 + ... + A_n + A_{n+1}$$

essentially this means if there are n arrivals in one unit time, the sum of interarrival time of the past n observations has to be less than or equal to one, but if one more interarrival time is added, it is greater then one (unit time).

- The A_is in the relation can be generated from uniformly distributed random number $A_i = (-1/\alpha)\ln R_i$, thus

$$\sum_{i=1}^{n} \frac{-1}{\alpha} \ln R_i \leq 1 < \sum_{i=1}^{n+1} \frac{-1}{\alpha} \ln R_i$$

both sides are multiplied by $-\alpha$

$$\ln \prod_{1}^{n} R_i = \sum_{i=1}^{n} \ln R_i \geq -\alpha > \sum_{i=1}^{n+1} \ln R_i = \ln \prod_{1}^{n+1} R_i$$

that is

$$\prod_{1}^{n} R_i \geq e^{-\alpha} > \prod_{1}^{n+1} R_i$$

- Now we can use the Acceptance-Reject method to generate Poisson distribution.

  Step 1. Set $n = 0$, $P = 1$.

  Step 2. Generate a random number R_{n+1} and replace P by $P * R_{n+1}$.

  Step 3. If $P < e^{-\alpha}$, then accept $N = n$, meaning at this time unit, there are n arrivals. Otherwise, reject the current n, increase n by one, return to Step 2.

- Efficiency: How many random numbers will be required, on the average, to generate one Poisson variate, N? If $N = n$, then $n+1$ random numbers are required (because of the (n+1) random numbers product).

$$E(N + 1) = \alpha + 1$$

- Example 9.10 on page 346, Example 9.11 on page 347
- When α is large, say $\alpha \leq 15$, the acceptance-rejection technique described here becomes too expensive. Use normal distribution to approximate Poisson distribution. When α is large

$$Z = \frac{N - \alpha}{\sqrt{\alpha}}$$

is approximately normally distributed with mean 0 and variance 1, thus

$$N = \lceil \alpha + \sqrt{\alpha} Z - 0.5 \rceil$$

can be used to generate Poisson random variate.

Chapter 9

Input Modeling

Collect and analyze input data so that the simulation can be fed with appropriate data. This is a very important task. Without proper input data, the good simulation model won't generate correct, appropriate result.

9.1 Identifying the Distribution with Data

9.1.1 Histograms

- A frequency distribution or histogram is useful in identifying the shape of a distribution.
- A histogram is constructed as follows.
  1. Divide the range of the data into intervals (intervals are usually of equal width).
  2. Label the horizontal axis to conform to the intervals selected.
  3. Determine the frequency of occurrences within each interval.
  4. Label the vertical axis so that the total occurrences can be plotted for each interval.
  5. Plot the frequencies on the vertical axis.
- The issue of intervals being too ragged, too coarse ... See Figure 10.1 on page 360.
- Example 10.2 on page 359, Example 10.3 on page 360

9.1.2 Selecting the Family of Distribution

There are many different distributions that may fit into a specific simulation task. Though exponential, normal and Poisson distributions are the ones used most often, others such as gamma and Weibull distributions are useful and important as well.

Here is a list of commonly used distributions.

Binomial Models the number of successes in n trials, when the trials are independent with common success probability, p.

Negative Binomial including the geometric distribution Models the number of trials required to achieve k successes.

Poisson Models the number of independent events that occur in a fixed amount of time or space.

Normal Models the distribution of a process that can be thought of as the sum of a number of component processes.

Log-normal Models the distribution of a process that can be thought of as the product of a number of component processes.

Exponential Models the time between independent events, or a process time which is memoryless.

Gamma An extremely flexible distribution used to model non-negative random variables.

Beta An extremely flexible distribution used to model bounded random variables.

Erlang Models processes that can be viewed as the sum of several exponentially distributed processes.

Weibull Models the time-to-failure for components.

Discrete or Continuous Uniform Models complete uncertainty, since all outcomes are equally likely.

Triangular Models a process when only the minimum, most-likely, and maximum values of the distribution are known.

Empirical Resamples from the actual data collected.

9.1.3 Quantile-Quantile Plots

- If X is a random variable with cdf F, then the q-quantile of X is the value γ such that $F(\gamma) = P(X \leq \gamma) = q$ for $0 < q < 1$.
- When F has an inverse, we write $\gamma = F^{-1}(q)$

- Now we discuss how to use Q-Q plots to draw distributions.
  - Let $\{x_i, i = 1, 2, ...n\}$ be a sample data from X.
  - Re-order the x_is as y_is such that $y_1 \leq y_2 \leq ... \leq y_n$.
  - The q-q plot is based on the fact that y_i is an estimate of the *(j-1/2)/n* quantile of X, that is,
  $$y_i \approx F^{-1}(\frac{j - \frac{1}{2}}{n})$$
  where F is the distribution function under the consideration that fits the samples.
  - Plot y_i against $F^{-1}(\frac{j-\frac{1}{2}}{n})$. If the distribution fits, the line would form a straight line. Use the measurement of deviation to decide whether or not the set of samples fit the distribution function.
- Example 10.4 on page 365.

9.1.4 Parameter Estimation

- After a family of distribution has been selected such as Poisson, Normal, Geometric ..., the next step is to estimate the parameters of the distribution.
- Sample mean and sample variance can be used to estimate the parameters in a distribution.
  - Let $X_1, X_2, ..., X_n$ be the sample of size n.
  - The sample mean is
  $$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$
  - The sample variance is
  $$S^2 = \frac{\sum_{i=1}^{n} {X_i}^2 - n\overline{X}^2}{n-1}$$
  - If the data are discrete and grouped in a frequency distribution, then we can re-write the equations as
  $$\overline{X} = \frac{\sum_{j=1}^{k} f_j X_j}{n}$$
  and
  $$S^2 = \frac{\sum_{j=1}^{k} f_j {X_j}^2 - n\overline{X}^2}{n-1}$$
  - Example 10.5 on page 368

- If the data are continuous, we "discretize" them and estimate the mean

$$\overline{X} \approx \frac{\sum_{j=1}^{c} f_j m_j}{n}$$

  and the variance

$$S^2 \approx \frac{\sum_{j=1}^{c} f_j {m_j}^2 - n\overline{X}^2}{n-1}$$

  where f_j is the observed frequency in the jth class interval, m_j is the midpoint of the jth interval, and c is the number of class intervals.

- Example 10.6 on page 369

- A few well-established, suggested estimators are listed in Table 10.3 on page 370, followed by examples. They come from theory of statistics.
- The examples include Poisson Distribution, Uniform Distribution, Normal Distribution, Exponential Distribution, and Weibull Distribution.

9.2 Goodness-of-Fit Tests

- Earlier we introduced hypothesis test to examine the quality of random number generators. Now we will apply these tests to hypothesis about distributional forms of input data.
- Goodness-of-fit tests provide helpful guidance for evaluating the suitability of a potential input model.
- The tests depends heavily on the amount of data. If very little data are available, the test is unlikely to reject *any* candidate distribution (because not enough evidence to reject); if a lot of data are available, the test will likely reject *all* candidate distributions (because none fits perfectly).
- Failing to reject a candidate should be viewed as a piece of evidence in favor of that choice; while rejecting an input model is only one piece of evidence against the choice.
- Chi-square test is for large sample sizes, for both discrete and continuous distributional assumptions, when parameters are estimated by maximum likelihood.
  - Arranging the n observations into a set of k class intervals or cells.
  - The test statistic

$${x_0}^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

    where O_i is the observed frequency in the ith class interval and E_i is the expected frequency in that class interval.

- The $x_0{}^2$ approximately follows the chi-square distribution with *k-s-1* degrees of freedom, where s represents the number of parameters of the estimated distribution. E.g Poisson distribution has s = *1*, normal distribution has *s=2*.
- The hypothesis

  H_0 : the random variable, X, conforms to the distributional assumption with the parameter(s) given by the parameter estimate(s)

  H_1 : the random variable X does not conform the distribution
- The critical value $x_{\alpha,k-s-1}{}^2$ is found in Table A.6. H_0 is rejected if $x_0{}^2 > x_{\alpha,k-s-1}{}^2$.
- The choice of k, the number of class intervals, see Table 10.5 on page 377.
- Example 10.13 on page 377.

- Chi-square test with equal probabilities:
  - If a continuous distributional assumption is being tested, class intervals that are equal in probability rather than equal in width of interval should be used.
  - Example 10.14: Chi-square test for exponential distribution (page 379)
    * test with intervals of equal probability (not necessary equal width)
    * number of intervals less than or equal to *n/5*
    * n = *50*, so $k \leq 10$, according to recommendations in Table 10.5, 7 to 10 class intervals be used.
    * Let k = *8*, thus $p = 0.125$
    * The end points for each interval are computed from the cdf for the exponential distribution

      $$F(a_i) = 1 - e^{-\lambda a_i}$$

      where a_i represents the end point of the ith interval.
    * Since $F(a_i)$ is the cumulative area from zero to a_i, thus $F(a_i) = ip$

      $$ip = 1 - e^{-\lambda a_i}$$

      thus

      $$a_i = -\frac{1}{\lambda}\ln(1 - ip) \quad i = 0, 1, ..., k$$

      regardless the value of λ, $a_0 = 0$ and $a_k = \infty$.
    * With $\lambda = 0.084$ in this example and k = *8*,

      $$a_1 = -\frac{1}{0.084}\ln(1 - 0.125) = 1.590$$

      continue with i = *2,3,...,7* results in 3.425, 5.595, 8,252, 11.677, 16.503, and 24.755.
    * See page 379 and 380 for completion of the example.
  - Example 10.15 (Chi-square test for Weibull distribution) on page 380

- Example 10.16 (Computing intervals for the normal distribution) on page 381
  * For the given data, using suggested estimator in Table 10.3 on page 370, we know (the original data was from Example 10.3 on page 360)

$$\mu = \bar(x) = 11.90$$

$$\sigma^2 = S^2 =$$

- Kolmogorov-Smirnov Goodness-of-fit test
  * Chi-square test heavily depends on the class intervals. For the same data, different grouping of the data may result in different conclusion, rejection or acceptance.
  * The K-S goodness-of-fit test is designed to overcome this difficulty. The idea of K-S test is from q-q plot.
  * The K-S test is particularly useful when sample size are small and when no parameters have been estimated from the data.
  * Example 10.7 on page 383, using the method described in Section 8.4.1 on page 299. A few notes:
    - If the interarrival time is exponentially distributed, the arrival times are uniformly distributed on (0,T]

Chapter 10

Verification and Validation of Simulation Models

So far, we have discussed how to run simulation, how to generate and test random number generators, how to build input data models. This chapter discusses how one might verify and validate a simulation model.

The gola of validation process

- to produce a model that represents true system behavior close enough for the model to be used as a substitute for the physical system
- to increase to an accetable level of credibility of the model

The verification and validation process consists of the following components.

1. Verification is concerned with building the model right.
2. Validation is concerned with building the right model.

10.1 Model Buidling, Verification, and Validation

Steps involved

- Observe the real system and the interaction among its various components, and collect data on its behavior.

- Construct a conceptual model – a collection of assumptions on the components and the structure of the system, plus hypotheses on the values of model input parameters.
- Translate the conceptual model into a computerized model.

10.2 Verification of Simulation Models

- Make sure the conceptual model is translated into a computerized model properly and truthfully. Typically three intuitive mechanisms can be used.
  1. Common sense
     (a) Have the computerized representation checked by someone other than its developer.
     (b) Make a flow diagram which includes each logically possible action of a system can take when an event occurs, and follow the model logic for each action for each event type.
     (c) Closely examine the model output under a variety of settings of input parameters.
     (d) Interactive Run Controller (IRC) or debugger is an essential component of successful simulation model building, which allows the analysist or the programmer to test the simulation program interactively.
     (e) If one is modeling certain queueing system, the simulation results can be compared with theoretical results as upper or lower bound. For example, in a queueing system, if the increase in arrival rate or service time does not affect the queue length or waiting time, then something is suspicious.
  2. Thourough documentation: by describing in detail what the simulation programmer has done in design and implementing the program, one can often identify some problems.
  3. Trace: go through a few steps in an actual program to see if the behavior is reasonable (Example 11.1 on page 404)
- Verification of a simulation model is very similar in what a typical software product would have to go through.

10.3 Calibration and Validation of Models

- For a given program, while common sense verification is possible, strict verification of a model is intractable, very much similar to the proof of correctness of a program.
- Validation is a process of comparing the model and its behavior to the real system and its behavior.
- Calibration is the iterative process of comparing the model with real system, revising the model if necessary, comparing again, until a model is accepted (validated).

- Three-step approach in the validation process.
  1. Build a model that has high face validity.
  2. Validate model assumptions
  3. Compare the model input-output transformations to corresponding input-output transformation for the real system.

  The three-step approach is discussed next.

10.3.1 Face Validity

- Build a "reasonable model" on its face to model users who are knowledgeable about the real system being simulated.
- Do some "sanity check"

10.3.2 Validation of Model Assumptions

- Model assumptions fall into two categories: structural assumptions and data assumptions.
- Structural assumptions deal with such questions as how the system operates, what kind of model should be used, queueing, inventory, reliability, and others.
- Data assumtions: what kind of input data model is? What are the parameter values to the input data model?

10.3.3 Validating Input-Output Transformations

- View the model as a black box
- Feed the input at one end and exmaine the output at the other end
- Use the same input for a real system, compare the output with the model output
- If they fit closely, the black box seems working fine
- Otherwise, something is wrong

The bank example (Example 11.2 on page 411)

10.3.4 Input-Output Validation: Using Historical Input Data

- In the previous bank example and other places, we used some artificial data as input to a simulation model.
- An alternative is to use past data as input (historical input data). Sometimes this is called *trace-driven* simulation.
- To conduct a validation test using historical input data, it is important that all the input data and all the system response data be collected during the same time period.
- Example 11.3 (The Candy Factory)
  - The Candy Factory has three machines, the Candy maker, the Candy packer, and the Box maker.
  - The three machines are connected by two conveyors.
  - The machines have random break-down time.
  - When a machine breaks down, it causes its incoming conveyors to be full and out-going conveyors to be empty.
  - The time-to-failure (operating time) and down-time of each machine is recorded. For machine i
  $$T_{i1}, D_{i1}, T_{i2}, D_{i2}, ...$$
  - The system responses: the production level Z_1, the number Z_2 and the time of occurence Z_3 of operator interventions were recorded.
  - The model responses ($Y_i, i = 1, 2, 3$) for the same parameters were collected for comparison to the corresponding system responses.
  - The differences between Y_i and Z_i can be used to build test statistics (in this case the Student Test t) to see if the model is statistically the same as the real system.
  - If there is only one set of input data, only one set of output can be obtained. The result doesn't carry much statistical significance.
  - If K sets of input data are available, K runs of tests can be conducted. See Table 11.6 for an example.
  - Let $Z_{ij}, j = 1, 2, ...K$ be the system responses and $W_{ij}, j = 1, 2, ...K$ be the model responses, the differences $d_j = Z_{ij} - W_{ij}$ are approximately normally distributed with mean μ_d and variance σ_d^2
  - If the mean μ is zero, then statistically W_{ij} are the same as Z_{ij}. A t test is conducted
  $$H_0 : \mu_d = 0$$
  and
  $$H_1 : \mu_d \neq 0$$

- The t statistic is computed as

$$t_0 = \frac{\overline{d} - \mu_d}{S_d/\sqrt{K}}$$

- The critical value $t_{\alpha/2,K-1}$ where K is the degree of freedom. In our example, K is the number of input data sets (thus, the number of experiments run).
- If $|t_0| \leq t_{\alpha/2,K-1}$ do not reject H_0, otherwise reject H_0.

- Example 11.4 (The Candy Factory continued) Use actual numbers (e.g. K = 5, etc.) to show how the above example works.

Chapter 11

Output Analysis for a Single Model

- Output analysis is the analysis of data generated by a simulation.
- How do we know if the result of a simulation is statistically significant?

11.1 Stochastic Nature of Output Data

- Simualtion result is generated for a given input data. Essentially a model is an input/output transformation.
- Since the input variables are random variables, the output variables are random variables as well, thus they are stochastic (probabilistic).
- Example 12.1 (Able and Baker, revisited)
  - Instead of a single run of simulation, four runs were conducted. The results are shown in Table 12.1 on page 431.
  - The utilization $\hat{\rho}_r$ and system time $\hat{w}_r$ were listed as 0.808,0.875,0.708, 0.842, and 3.74, 4.53, 3.84, 3.98.
  - There are two general questions we have to address by a statistical analysis of the observed utilization $\hat{\rho}_r$
    1. Estimation of the true utilization $\rho = E(\hat{\rho}_r)$ by a single value, called a point estimate.
    2. Estimation of the error in our point estimate, either in the form of a standard error or confidence interval. This is called an interval estimate.
- Example 12.2 (Effect of correlation and initial condition) We will see actual measurement of performance in Section 12.3 and 12.4.
- Example 12.3 (Fifth National Bank of Jasper, revised)

11.2 Types of Simulations with Respect to Output Analysis

- Terminating simulation and steady-state simulations:
  - A *terminating* simulation is one that runs for some duration of time T_E, where E is a specified event or set of events which stops the simulation.
  - A *steady-state* simulation is a simulation whose objective is to study long-run, or steady-state, behavior of a non-terminating system.
- Examples of terminating simulation (Example 12.4 - 12.7) on pages 435-436.
- Examples of steady-state simulation (Example 12.8 - 12.9) on page 437.

11.3 Measures of Performance and Their Estimation

Consider a set of output values for the same measure $Y_1, Y_2, ..., Y_n$ (e.g. delays of n different runs, or waiting times of n different runs). We want to have

- a point estimate to approximate the true value of Y_i, and
- an interval estimate to outline the range where the true value lies.

11.3.1 Point Estimation

- The point estimator of θ based on the data $Y_1, Y_2, ...Y_n$ is defined by

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

- The point estimator $\hat{\theta}$ is said to be unbiased for θ if

$$E(\hat{\theta}) = \theta$$

  In general

$$E(\hat{\theta}) = \theta + b$$

  i.e. there is a drifting, or bias.

- For continuous data, the point estimator of ϕ based on data $\{Y(t), 0 \leq t \leq T_E\}$, where T_E is the simulation run length, is defined by

$$\hat{\phi} = \frac{1}{T_E}\int_0^{T_E} Y(t)dt$$

  and is called a time average of *Y(t)* over $[0, T_E]$.

- In general
$$E(\hat{\phi}) = \phi + b$$
if $b = 0$, $\hat{\phi}$ is said to be unbiased for ϕ

- One performance measure of these estimators (point or interval) is a quantile or a percentile.
  - Quantiles describe the level of performance that can be delivered with a given probability p
  - Assume Y represents the delay in queue a customer experiences, then the 0.85 quantile (or 85% percentile) of Y is the value γ such that
  $$P(Y \leq \gamma) = p = 0.85$$

11.3.2 Interval Estimation

- Valid interval estimation typically requires a method of estimating the variance of the point estimator $\hat{\theta}$ or $\hat{\phi}$.
- Let $\sigma^2(\hat{\theta}) = var(\hat{\theta})$ represent the true variance of a point estimator $\hat{\theta}$, and let $\hat{\sigma^2}(\hat{\theta})$ represent an estimator of $\sigma^2(\hat{\theta})$ based on the data $Y_i, i = 1, 2, ..., n$.
- Suppose that
$$E[\hat{\sigma^2}(\hat{\theta})] = B\sigma^2(\hat{\theta})$$
where B is called the bias in the variance estimator.
- It is desirable to have
$$B = 1$$
in which case $\hat{\sigma^2}(\hat{\theta})$ is said to be an unbiased estimator of variance, $\sigma^2(\hat{\theta})$.
- If it is an unbiased estimator, the statistic
$$t = \frac{\hat{\theta} - \theta}{\hat{\sigma}(\hat{\theta})}$$
is approximatly t distribution with some degree f of freedom.
- An approximate $100(1-\alpha)\%$ confidence interval for θ is given by
$$\hat{\theta} - t_{\alpha/2,f}\hat{\sigma}(\hat{\theta}) \leq \theta \leq \hat{\theta} + t_{\alpha/2,f}\hat{\sigma}(\hat{\theta})$$
- This relation involves three parameters, estimator for mean, estimator for variance, and the degree of freedom. How to determine these values?

– Estimator for mean is calculated as above as a point estimator

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

– Estimator for the variance and for the degree of freedom has to consider two separate cases

1. If Y_is are statistically independent observations then use

$$S^2 = \sum_{i=1}^{n} \frac{(Y_i - \hat{\theta})^2}{n-1}$$

to calculate

$$\hat{\sigma}^2(\hat{\theta}) = \frac{S^2}{n}$$

with the degree of freedom $f = n - 1$.

2. If Y_is are not statistically independent, then the above estimator for variance is biased. Y_is is an autocorrelated sequence, sometimes called a time series.
In this case,

$$\sigma^2(\hat{\theta}) = var(\hat{\theta}) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} cov(Y_i, Y_j)$$

i.e. one needs to calculate co-variance for every possible pair of observations. Too expensive.
If the simulation is long eough to have passed the transient phase, the output is approximately *covariance stationary*. That is Y_{i+k} depends on Y_i in the same way as Y_k depends on Y_1
For a covariance stationary time series Y_is, define the lag k autocovariance by

$$\gamma_k = cov(Y_1, Y_{1+k}) = cov(Y_i, Y_{i+k})$$

For k = 0, γ_0 becomes the population variance

$$\gamma_0 = cov(Y_{0+i}, Y_i) = var(Y_i)$$

The lag k autocorrelation is the correlation between any two observations k apart.

$$\rho_k = \frac{\gamma_k}{\gamma_0}$$

and has the property

$$-1 \leq \rho_k \leq 1, \quad k = 1, 2, ...$$

If a time series is covariance stationary, then the calculation of sample variance can be substantially simplified.

$$\sigma^2(\hat{\theta}) = \frac{\gamma_0}{n}\left[1 + 2\sum_{k=1}^{n-1}\left(1 - \frac{k}{n}\right)\rho_k\right]$$

So all we need is to calculate covariance between one sample and every other samples, but not every sample with every other samples.

Some discussions about why autocorrelation make it difficult to estimate $\sigma^2(\hat{\theta})$ are skipped

11.4 Output Analysis for Terminating Simulations

- Use independent replications, i.e. the simulation is repeated a total of R times, each run using a different random number stream and independently chosen initial conditions.
- Let Y_{ri} be the ith observations within replication r, for $i = 1, 2, ...n_r$ and r = *1, 2, ..., R*.
- For a fixed r, $Y_{r1}, Y_{r2}, ...$ is an autocorrelated sequence. For different replications $r \neq s$ Y_{ri} and Y_{sj} are statistically independent.
- Define a sample mean

$$\hat{\theta_r} = \frac{1}{n_r}\sum_{i=1}^{n_r} Y_{ri}, \quad r = 1, 2, ..., R$$

- There are R samples, so R sample means, the overall point estimate is

$$\hat{\theta} = \frac{1}{R}\sum_{r=1}^{R} \hat{\theta}_r$$

11.4.1 Interval Estimate for a Fixed Number of Replications

- The sample mean can be calculated as above
- The sample variance

$$\hat{\sigma}^2(\hat{\theta}) = \frac{1}{R}\sum_{r=1}^{R} \frac{(\hat{\theta_r} - \hat{\theta})^2}{R-1}$$

- When the output data are of the form $\{Y_r(t), 0 \leq t \leq T_E\}$ for r = *1, 2, ..., R*

$$\hat{\phi_r} = \frac{1}{T_E}\int_0^{T_E} Y_r(t)dt, \quad r = 1, 2, ..., R$$

$$\hat{\phi} = \frac{1}{R}\sum_{r=1}^{R} \hat{\phi_r}$$

and

$$\hat{\sigma}^2(\hat{\phi}) = \frac{1}{R}\sum_{r=1}^{R} \frac{(\hat{\phi_r} - \hat{\phi})^2}{R-1}$$

- The sample variance is essentially a measure of error. In statistics, we use its square root as the *standard error*.
- Example 12.10 on page 446 (The Able-Baker carhop problem, continued): calculate the confidence interval for Able's utilization and average waiting time.
- Example 12.11 on page 447: the more runs, the more accurate the result is.

11.4.2 Interval Estimate with Specified Precision

- Previous section discusses for a given set of replications to calcualte the confidence interval and error. Sometimes we need to do the inverse, given a level of error and confidence, how many replications are needed?
- The half-length (h.i.) of a $100(1-\alpha)\%$ confidence interval for a mean θ, based on the t distribution, is

$$h.i. = t_{\alpha/2,R-1} * \hat{\sigma}(\hat{\theta})(*)$$

  where $\hat{\sigma}(\hat{\theta}) = S/\sqrt{R}$, S is the sample standard deviation, R is the number of replications.
- Assume an error criterion ϵ is specified with a confidence level $1-\alpha$, it is desired that a sufficiently large sample size R be taken such that

$$P(|\hat{\theta} - \theta| < \epsilon) \geq 1 - \alpha$$

- Since we have the relation (*), the desired the error control condition can be written as

$$h.i. = \frac{t_{\alpha/2,R-1}S_0}{\sqrt{R}} \leq \epsilon$$

- Solve the above relation, we have

$$R \geq \left(\frac{t_{\alpha/2,R-1}S_0}{\epsilon}\right)^2$$

  since $t_{\alpha/2,R-1} \geq z_{\alpha/2}$ the above relation can be written

$$R \geq \left(\frac{z_{\alpha/2}S_0}{\epsilon}\right)^2$$

  For $R \geq 50, t_{\alpha/2,R-1} \approx z_{\alpha/2}$ the inequality with standard normal distribution holds.

  This says we need to run that many (R) replications to satisfy the error requirement.
- The true value of θ is in the following range with probability of $100(1-\alpha)\%$

$$\hat{\theta} - \frac{t_{\alpha/2,R-1}S}{\sqrt{R}} \leq \theta \leq \hat{\theta} + \frac{t_{\alpha/2,R-1}S}{\sqrt{R}}$$

- Example 12.12 on page 449

11.5 Output Analysis for Steady-State Simulations

- Consider a single run of a simulation model whose purpose is to estimate a *steady state*, or *long run*, characteristics of the system.
- Assume $Y_1, Y_2, ...$ are observations, which in general are samples of an autocorrelated time series.
- The steady-state measure of performance θ to be estimated is defined by

$$\theta = lim_{n\rightarrow\infty}\frac{1}{n}\sum_{i=1}^{n}Y_i$$

- This result is independent of initial conditions, random number streams, ...

11.5.1 Initialization Bias in Steady-State Simulations

There are several methods of reducing the point estimator bias which is caused by using artificial and unrealistic initial conditions in a steady-state simulation.

1. Initialize the simulation in a state that is more representative of long-run conditions. E.g. use a set of real data as initial condition.
2. Divide the simulation into two phases, warm-up phase and steady state phase. Data collection doesn't start until the simulation passes the warm-up phase.

Consider the example on page 452 (Example 12.13)

- A set of 10 independent runs, each run was divided into 15 intervals. The data were listed in Table 12.5 on page 453.
- Typicall we calculate average *within* a run. Since the data collected in each run is most likely autocorrelated, a different method is used to calculate the average *across* the runs.
- Such averages are known as *ensemble average*.

Several issues:

1. Ensemble average will reveal a smoother and more precise trend as the number of replications, R, is increased.
2. Ensemble average can be smoothered further by plotting a *moving average*. In a moving average each plotted point is actually the average of several adjacent ensemble averages.

3. Cumulative averages become less variable as more data are averaged. Thus, it is expected that the curve at left side (the starting of the simulation) of the plotting is less smooth than the right side.
4. Simulation data, especially from queueing models, usually exhibits positive autocorrelation. The more correlation present, the longer it takes for the average to approach steady state.
5. In most simulation studies the analyst is interested in several measures such as queue length, waiting time, utilization, etc. Different performance measures may approach stead state at different rates. Thus it is important to examine each performance measure individidually for initialization bias and use a deletion point that is adequate for all of them.

11.5.2 Replication Method for Steady-State Simulations

- If initialization bias in the point estimator has been reduced to a negligible level, the method of independent replications can be used to estimate point-estimator variability and to construct a confidence interval.
- If significant bias remains in the point estimator and a large number of replications are used to reduce point estimator variability, the result confidence interval can be misleading.
  - The bias is not affected by the number of replications R, but by deleting more data (i.e. increasing T_0) or extending the length of each run (i.e. increasing T_E.
  - Increasing the number of replications R may produce shorter confidence intervals around a "wrong point" $\theta + b$, rather than θ
- If d is the number of observations to delete from a total of n observations, a rough rule is n-d should be at least *10d*, or T_E should be at least $10T_0$.
- Given the run length, the number of replications should be as many as possible. Kelton in 1986 established that there is little value to run more than 25 replications. So if time is available, make the simulation longer, instead of making more replications.
- See Example 12.14 on page 460 and Example 12.15 on page 461, where Example 12.15 demonstrate the cases where few obsevations were deleted.

  A couple of notes as fewer observations were deleted (d is smaller):

  1. The confidence interval shifts downward, reflecting the greater downward bias in $\overline{Y}_{..}(15, d)$ as d decreases. This can be attributed as the result of a "cold start".
  2. The standard error of $\overline{Y}_{..}(15, d)$, namely $S/\sqrt{R}$ decreases as d decreases. As d decreases, the number of samples included in the statistics increases, reducing the error range.

11.5.3 Sample Size in Steady-State Simulations

In a steady-state simulation, a specified precision may be achieved either by increasing the number of replications R, or by increasing the run length T_E.

Example 12.16 on page 462 shows an example of calculating R for a given precision.

Example 12.17 on page 463 shows an example of increasing T_E for the given precision (recall the general rule: T_E should be at least $10T_0$).

11.5.4 Batch Means for Interval Estimation in Steady-State Simulations

- One disadvantage of replication is that data must be deleted on each replication.
- One disadvantageof a single-replication is its data tend to be autocorrelated.
- The method of *batch mean* divides the output data from one replication into a few large batches.
- Treat the means of these batches as if they were independent. See the formula on page 463 and 464.
- The key isssue is that no commonly accepted method for choosing an accetable batch size m. This is actually one of the research areas in simulation.
  - Schmeiser found for a *fixed total sample size* there is little benefit from dividing it into more than $k = 30$ batches.
  - Although there is typically autocorrelation between batch means at all lags, the lag-1 autocorrelation $corr(\hat{Y_j}, \hat{Y_{j+1}})$ is usually studied to assess the dependence between batch means.
  - The lag-1 autocorrelation between batch means can be estimated using the method described earlier. They should not be estimated from a small number of batch means, i.e. we need to have large number of batches, though the size of batches could be small.
  - If the total sample size is to be chosen sequentially (i.e. choose one for one experiment, choose another one for improvement etc.), then it is helpful to allow the batch size and number of batches to grow as the run length increases.
- Example 12.18 on page 465.