Lecture Notes of

Modeling and Simulation
7th Sem IT

BCS-408 **<u>MODELING & SIMULATION (3-1-0)</u>** **Cr.-4**

Module I **(10 Lectures)**

Inventory Concept: The technique of Simulation. : 1 class
Major application areas, concept of a System. : 1 class
Environment. : 1 class
Continuous and discrete systems. : 1 class
Systems modeling, types of models. : 1 class
Progress of a Simulation Study. : 1 class
Monte Carlo Method. : 1 class
Comparison of Simulation and Analytical Methods. : 1 class
Numerical Computation Technique for discrete and continuous models. : 1 class
Continuous System Simulation. ;1 class
Revision

Module II **(12 Lectures)**

Probability Concepts in Simulation: 2 classes
Stochastic variables, Discrete and Continuous Probability Functions. 2 classes
Numerical evaluation of continuous probability functions, continuous uniformly distributed random numbers. : 2 classes
Random Number Generators – Linear congruential Generator, Mid Square Method, Multiplicative Congruential generator, rejection Method. : 2 classes
Testing of random Numbers. : 2 classes
Generation of Stochastic variants. : 1 class
Arrival Patterns Service times. : 1 class
Revision

Module III **(10 Lectures)**

Discrete System Simulation and GPSS: Discrete Events, Representation of Time, generation of arrival patterns. : 2 classes
Fixed time step versus next event simulation, Simulation of a
Telephone System, Delayed calls . : 2 classes
Introduction to GPSS: Creating and moving transactions, queues. : 2 classes
Facilities and storages, gathering statistics, conditional transfers, program control statements, priorities and parameters. : 2 classes
Standard numerical attributes, functions, gates, logic switches and tests, Variables, Select and Count. : 2 classes
Revision

Module IV **(10 Lectures)**

Simulation Languages and Practical Systems : 1 class
Continuous and discrete systems languages, factors in the section of discrete systems simulation language. ; 2 classes
Computer model of queuing, inventory and scheduling systems. : 2 classes
Design and Evaluation of simulation Experiments: Length of simulation runs, validation, variance reduction techniques. : 2 classes
Experimental layout, analysis of simulation output,
Recent trends and developments. : 1 class
Revision

Books:

1. System Simulation – Geoffrey Gordon, 2nd Edition, PHI
2. System Simulation with Digital computer – Narsingh Deo, PHI

—

Module-I

Objectives:

- To give an overview of the course (Modeling & simulation).
- Define important terminologies.
- Classify systems/models

System:
any set of interrelated components acting together to achieve a common objective.

Examples:

1. **Battery**
   - Consists of anode, cathode, acid and other omponents.
   - These components act together to achieve one objective like preserving electricity.
2. **University**
   - Consists of professors, students and employees.
   - These objects act together to achieve the objective of teaching & learning process.

A system consists of

- **Inputs**
  Elements that cause changes in the systems variables.
- **Outputs**
  Response
- **Systems (process)**

Defines the relationship between the inputs and outputs

Some Possible Inputs

- Inlet flow rate
- Temperature of
entering material

- Concentration of entering material

Some Possible Outputs

- Level in the tank
- Temperature of material in tank
- Outlet flow rate
- Concentration of material in tank

Qn: What inputs and outputs are needed when we want to model the Inventory Control System?

Model: A model describes the mathematical relationship between inputs and outputs.

Simulation: is the process of using the mathematical model to determine the response of the system in different situations in a Computer system.

Classification of Systems

Systems can be classified based on different criteria:

- Spatial characteristics: lumped & distributed
- Continuity of the time variable: Continuous, discrete-time
- Quantization of dependent variable: Quantized & Non-quantized
- Parameter variation: time varying & fixed (time-invariant)
- Superposition principle: linear & nonlinear

Continuous-time System:

- The signal is defined for all t in an interval [ti, tf]

Discrete-time System:

- The signal is defined for a finite number of time points {t0, t1,…}

A system is linear:

- if it satisfies the **super position principle.**
- A system satisfies the superposition principle if the following conditions are satisfied:
  1. Multiplying the input by any constant, multiplies the output by the same constant.
  2. The response to several inputs applied simultaneously is the sum of individual response to each input applied separately.

Quantized variable System:
- The variable is restricted to a finite or countable number of distinct values.

Non-Quantized variable System:
- The variable can assume any value within a continuous range.

Characteristics of Lumped Systems:
- Only one independent variable (t)
- No dependence on the spatial coordinates
- Modeled by ordinary differential equations
- Needs a finite number of state variables

Distributed System:
- More than one independent variable
- Depends on the spatial coordinates or some of them.
- Modeled by partial differential equations
- Needs an infinite number of state variables.

Time variant Systems:
- Observes the change of state of the variable regularly and records the related information at that point of change.

Fixed-Time Event Systems:
- Changes of the state of the variable occurs at some constant interval.

Models and types:
Models are the replica of systems which can be represented physically or mathematically.
All the physical and mathematical models can further be divided into categories like: static and dynamic.

Simulation models may be either deterministic or stochastic (meaning probabilistic). In a **deterministic** simulation, all of the events and relationships among the variables are governed entirely by a combination of known, but possibly complicated, rules. The advantage of simulation is that you can still answer the question even if the model is too complicated to solve analytically.
In a **stochastic** simulation, ‘‘random variables’’ are included in the model to represent the influence of factors that are unpredictable, unknown, or beyond the scope of the model we use in the simulation.
.
In many applications, such as a model of the tellers in a bank, it makes sense to incorporate random variables into the model. In the case of a bank, we might wish to assume that there is a stream of anonymous customers coming in the door at unpredictable times, rather than explicitly modeling the behavior of each of their actual customers to determine when they plan to go to the bank.
It is worth noting here that it is well known in statistics that when we combine the actions of a large population of more-or-less *independently* operating entities (customers, etc.) the resulting behavior appears to have been randomly produced, and that the patterns of activity followed by the individual entities within that population are unimportant. For example, all of the telephone systems designed in the last 60 years are based on the (quite justified) assumption that the umber

of calls made in fixed length time intervals obeys the Poisson distribution. Thus the generation and use of random variables is an important topic in simulation.

Static Versus Dynamic Simulation Models

Another dimension along which simulation models can be classified is that of time. If the model is used to simulate the operation of a system over a period of time, it is **dynamic**. The baseball example above uses dynamic simulation. On the other hand, if no time is involved in a model, it is **static**. Many gambling situations (e.g., dice, roulette) can be simulated to determine the odds of winning or losing. Since only the number of bets made, rather than the duration of gambling, matters, static simulation models are appropriate for them

Monte Carlo Simulation (named after a famous casino town1 in Europe) refers to the type of simulation in which a static, approximate, and stochastic model is used for a deterministic system.

Let us now look at an example of Monte Carlo simulation. Consider estimating the value of π by finding the approximate area of a circle with a unit radius. The first quadrant of the circle is enclosed by a unit square.

If pairs of uniformly distributed pseudo-random values for the x and y coordinates in [0, 1] are generated, the probability of the corresponding points falling in the quarter circle is simply the ratio of the area of the quarter circle to that of the unit square.

The above program is an example of *Monte Carlo integration*, by which definite integral of arbitrary but finite functions can be estimated. Consider the following integral:

Such an integral can be partitioned into segments above or below the horizontal axis. Without loss of generality, let us assume $f(x) \geq 0$, $a \leq x \leq b$. We can then bound the curve with a rectangle with borders of $x = a$, $x = b$, $y = 0$, and $y = y$ max, where y max is the maximum value of $f(x)$ in the interval [a, b]. By generating random points uniformly distributed over this rectangular area, and deciding whether they fall above or below the curve $f(x)$, we can estimate the integral.

Dynamic Simulation

In the remainder of chapter, dynamic stochastic simulation models will be emphasized. A dynamic simulation program that is written in a general purpose programming language (such as Turing) must:

i) keep track of ‘‘simulated’’ time,

ii) schedule events at ‘‘simulated’’ times in the future, and

iii) cause appropriate changes in state to occur when ‘‘simulated’’ time reaches the time at which one or more events take place.

The structure of a simulation program may be either time-driven or event-driven, depending on the nature of the basic loop of the program. In **time-driven** models (see Figure 2a), each time through the basic loop, simulated time is advanced by some ‘‘small’’ (in relation to the rates of change of the variables in the program) fixed amount, and then each possible event type is checked to see which, if any, events have occurred at that point in simulated time. In **event-driven** models (see Figure 2b), events of various types are scheduled at future points in simulated time. The basic program loop determines when the next scheduled event should occur, as the minimum of the scheduled times of each possible event. Simulated time is then advanced to exactly that event time, and the corresponding event handling routine is executed to reflect the change of state that occurs as a result of that event.

Constructing a Simulation Model

1. **Identification of Components**

Our first task is to identify those system components that should be included in the model. This choice depends not only on the real system, but also on the aspects of its behavior that we intend to investigate. The only complete model of a system is the system itself. Any other model includes assumptions or simplifications. Some components of a system may be left out of the system model *if* their absence is not expected to alter the aspects of behavior that we wish to observe.

We assume that the bank automation problem specification includes the following information:

i) the times between successive arrivals of customers, expressed as a probability distribution,

ii) the distribution of the number of liters of gasoline needed by customers,

iii) the distribution of how long service at the pump takes as a function of the number of litres needed,

iv) the probability that an arriving customer will *balk* as a function of the number of cars already waiting

at the service station and the number of liters of gasoline he needs,

v) the profit per liter of gasoline sold, and

vi) the cost per day of running a pump (including an attendant's salary, etc.).

2. **Entities**

Customers, resources, service facilities, materials, service personnel, etc., are *entities*. Each type of entity has a set of relevant *attributes*. In our service station example, the entities are *cars,* with the attributes 'arrival time' and 'number of liters needed', and *pumps,* with the attribute 'time to the completion of service to the current customer'.

3. **Events**

Events are occurrences that alter the system state. Here the events are the *arrival* of a customer at the station, the *start-of-service* to a customer at a pump, and the *completion-of-service* to a customer at a pump. The first *arrival* event must be scheduled in the initialization routine; the remaining arrivals are handled by letting each invocation of the arrival routine schedule the next arrival event. The scheduling of a *start-of-service* event takes place either in the arrival routine, if there are no other cars waiting and a pump is available, or in the completion-of service routine, otherwise. Each time the start-of-service routine is invoked, the *completion-of-service* at that pump is scheduled.

Besides the types of events identified in the real system, there are two other **pseudo-events** that should be included in every simulation program. *End-of-simulation* halts the simulation after a certain amount of simulated time has elapsed, and initiates the final output of the statistics and measurements gathered in the run. End-of simulation should be scheduled during the initialization of an event-driven simulation; for time-driven simulations, it is determined by the upper bound of the basic program loop. A *progress-report* event allows a summary of statistics and measurements to be printed after specified intervals of simulated time have elapsed. These progress-report summaries can be used to check the validity of the program during its development, and also to show whether or not the system is settling down to some sort of ‘‘equilibrium’’ (i.e., stable) behavior.

4. **Groupings**

Similar entities are often *grouped* in meaningful ways. Sometimes an *ordering* of the entities within a group
is relevant. In the service station example, the groupings are *available* pumps (that are not currently serving any auto), *busy* pumps (ordered by the service completion time for the customer in service), and *autos* awaiting service (ordered by time of arrival). In a Turing program, such groupings can be represented as linked lists of records, as long as a suitable link field is included in the records for that entity type.

5. **Relationships**

Pairs of non-similar entities may be *related*. For example, a busy pump is related to the auto that is being served. Relationships can be indicated by including in the record for one of the entities in a pair a link to the other entity. In some cases, it may be desirable for each of the entities in the pair to have a link to the other entity. For example, the relationship between a busy pump and the auto being served can be represented by a link from the pump record to the corresponding auto record.

6. **Stochastic Simulation**

In a stochastic simulation, certain attributes of entities or events must be chosen ‘‘at random’’ from some probability distribution. In the service station example, these include the customer inter arrival times (i.e., the times between successive arrivals), the number of liters needed, the service time (based on the number of liters needed), and whether the customer decides to balk (based on the length of the waiting queue and the number of liters needed).

7. **Strategies**

Typically, a simulation experiment consists of comparing several alternative approaches to running the system to find out which one(s) maximize some measure of system ‘‘performance’’. In the case of the service station, the strategies consist of keeping different numbers of pumps in service.

8. **Measurements**

The activity of the system will be reflected in *measurements* associated with entities, events, groups, or relationships. Such measurements are used in specifying the performance of the system. If the measurement is associated with an entity, it should be recorded in a field contained within the record for that entity. For measurements associated with events, groupings or relationships, additional variables or records must be declared to record the measured quantities.
In our example system, we might wish to measure the numbers of customers served, liters of gasoline sold, customers who balk, and liters of gasoline *not* sold to balking customers, the total waiting time (from which we can calculate average waiting time), the total service time (to get pump utilization), and the total time that the waiting queue is empty.
conditions due to customers left in the system when end-of-simulation is reached are handled properly.

Advantages and disadvantages of simulation

Competition in the computer industry has led to technological breakthroughs that are allowing hardware companies to continually produce better products. It seems that every week another company announces its latest release, each with more options, memory, graphics capability, and power.
What is unique about new developments in the computer industry is that they often act as a springboard for other related industries to follow. One industry in particular is the simulation software industry. As computer hardware becomes more powerful, more accurate, faster, and easier to use, simulation software does too.
The number of businesses using simulation is rapidly increasing. Many managers are realizing the benefits of utilizing simulation for more than just the one-time remodeling of a facility. Rather, due to advances in software, managers are incorporating simulation in their daily

operations on an increasingly regular basis.
Advantages:
For most companies, the benefits of using simulation go beyond just providing a look into the future. These benefits are mentioned by many authors [Banks, Carson, Nelson, and Nicol, 2000; Law and Kelton, 2000; Pegden, Shannon and Sadowski, 1995; and Schriber, 1991]
and are included in the following:

Choose Correctly. Simulation lets you test every aspect of a proposed change or addition without committing resources to their acquisition. This is critical, because once the hard decisions have been made, the bricks have been laid, or the material-handling systems have been installed, changes and corrections can be extremely expensive. Simulation allows you to test your designs without committing resources to acquisition.

Time Compression and Expansion. By compressing or expanding time simulation allows you to speed up or slow down phenomena so that you can thoroughly investigate them. You can examine an entire shift in a matter of minutes if you desire, or you can spend two hours examining all the events that occurred during one minute of simulated activity.

Understand "Why?" Managers often want to know why certain phenomena occur in a real system. With simulation, you determine the answer to the "why" questions by reconstructing the scene and taking a microscopic examination of the system to determine why the phenomenon occurs. You cannot accomplish this with a real system because you cannot see or control it in its entirety.

Explore Possibilities. One of the greatest advantages of using simulation software is that once you have developed a valid simulation model, you can explore new policies, operating procedures, or methods without the expense and disruption of experimenting with the real system. Modifications are incorporated in the model, and you observe the effects of those changes on the computer rather than the real system.

Diagnose Problems. The modern factory floor or service organization is very complex. So complex that it is impossible to consider all the interactions taking place in one given moment. Simulation allows you to better understand the interactions among the variables that make up such complex systems. Diagnosing problems and gaining insight into the importance of these variables increases your understanding of their important effects on the performance of the overall system.
The last three claims can be made for virtually all modeling activities, queuing, linear programming, etc. However, with simulation the models can become very complex and, thus, have a higher fidelity, i.e., they are valid representations of reality.

Identify Constraints. Production bottlenecks give manufacturers headaches. It is easy to forget that bottlenecks are an effect rather than a cause. However, by using simulation to perform bottleneck analysis, you can discover the cause of the delays in work-in-process, information, materials, or other processes.

Develop Understanding. Many people operate with the philosophy that talking loudly, using computerized layouts, and writing complex reports convinces others that a manufacturing or service system design is valid. In many cases these designs are based on someone's thoughts about the way the system operates rather than on analysis. Simulation studies aid in providing understanding about how a system really operates rather than indicating an individual's predictions about how a system will operate.

Visualize the Plan. Taking your designs beyond CAD drawings by using the animation features offered by many simulation packages allows you to see your facility or organization actually running. Depending on the software used, you may be able to view your operations from various angles and levels of magnification, even 3-D. This allows you to detect design flaws that appear credible when seen just on paper on in a 2-D CAD drawing.

Build Consensus. Using simulation to present design changes creates an objective opinion. You avoid having inferences made when you approve or disapprove of designs because you simply select the designs and modifications that provided the most desirable results, whether it be increasing production or reducing the waiting time for service. In addition, it is much easier to accept reliable simulation results, which have been modeled, tested, validated, and visually represented, instead of one person's opinion of the results that will occur from a proposed design.

Prepare for Change. We all know that the future will bring change. Answering all of the "what-if" questions is useful for both designing new systems and redesigning existing systems. Interacting with all those involved in a project during the problem-formulation stage gives you an idea of the scenarios that are of interest. Then you construct the model so that it answers questions pertaining to those scenarios. What if an automated guided vehicle (AGV) is removed from service for an extended period of time? What if demand for service increases by 10 percent? What if....? The options are unlimited.

Wise Investment. The typical cost of a simulation study is substantially less than 1% of the total amount being expended for the implementation of a design or redesign. Since the cost of a change or modification to a system after installation is so great, simulation is a wise investment.

Train the Team. Simulation models can provide excellent training when designed for that purpose. Used in this manner, the team provides decision inputs to the simulation model as it progresses. The team, and individual members of the team, can learn by their mistakes, and learn to operate better. This is much less expensive and less disruptive than on-the-job learning.

Specify Requirements. Simulation can be used to specify requirements for a system design. For example, the specifications for a particular type of machine in a complex system to achieve a desired goal may be unknown. By simulating different capabilities for the machine, the requirements can be established.

B. Disadvantages

The disadvantages of simulation include the following:

Model Building Requires Special Training. It is an art that is learned over time and through experience. Furthermore, if two models of the same system are constructed by two competent individuals, they may have similarities, but it is highly unlikely that they will be the same.

Simulation Results May Be Difficult to Interpret. Since most simulation outputs are essentially random variables (they are usually based on random inputs), it may be hard to determine whether an observation is a result of system interrelationships or randomness.

Simulation Modeling and Analysis Can Be Time Consuming and Expensive.

Skimping on resources for modeling and analysis may result in a simulation model and/or analysis that is not sufficient for the task.

Simulation May Be Used Inappropriately. Simulation is used in some cases when an analytical solution is possible, or even preferable. This is particularly true in the simulation of some waiting lines where closed-form queueing models are available, at least for long-run evaluation.

Pseudo-Random Number Generation

Probability is used to express our confidence in the outcome of some random event as a real number between 0 and 1. An outcome that is impossible has probability 0; one that is inevitable has probability 1. Sometimes, the probability of an outcome is calculated by recording the outcome for a very large number of occurrences of that random event (the more the better), and then taking the ratio of the number of events in which the given outcome occurred to the total number of events.

It is also possible to determine the probability of an event in a non-experimental way, by listing all possible non-overlapping outcomes for the experiment and then using some insight to assign a probability to each outcome.

For example, we can show that the probability of getting ‘‘heads’’ twice during three coin tosses should be 3⁄8 from the following argument. First, we list all eight possible outcomes for the experiment (TTT, TTH, THT, THH, HTT, HTH, HHT, HHH). Next, we assign equal probability to each outcome (i.e., 1⁄8), because a ‘‘fair’’ coin should come up heads half the time, and the outcome of one coin toss should have no influence on another. (In general, when we have no other information to go on, each possible outcome for an experiment is assigned the same probability.)

And finally, we observe that the desired event (namely two H’s) includes three of these outcomes, so that its

probability should be the sum of the probabilities for those outcomes, namely 3⁄8.

A **random variable** is a variable, say X, whose value, x, depends on the outcome of some random event. For example, we can define a random variable that takes on the value of 1 whenever ‘‘heads’’ occurs and 0 otherwise.

Generating Uniform Pseudo-Random Numbers

Since the result of executing any computer program is in general both predictable and repeatable the idea that a computer can be used to generate a sequence of *random* numbers seems to be a contradiction. There is no contradiction, however, because the sequence of numbers that is actually generated by a computer is entirely predictable once the algorithm is known. Such algorithmically generated sequences are called **pseudo-random** sequences because they *appear* to be random in the sense that a good pseudo-random number sequence can pass most statistical tests designed to check that their distribution is the same as the intended distribution. On the other hand, to call them ‘‘random’’ numbers is no worse than it is to label floating point numbers as ‘‘real’’ numbers in a programming language.

It is worth noting that the availability of pseudo-random (rather than truly random) numbers is actually an *advantage* in the design of simulation experiments, because it means that our ‘‘random’’ numbers are *reproducible*.

Thus, to compare two strategies for operating our service station, we can run two experiments with different numbers of pumps but with exactly the same sequence of customers. Were we to attempt the same comparison using an actual service station, we would have to try the two alternatives one after the other, with a different sequence of customers (e.g., those that came on Tuesday instead of those that came on Monday), making it difficult to claim that a change in our

profits was due to the change in strategy rather than the differences in traffic patterns between Mondays and Tuesdays.
In general, most pseudo-random number generators produce uniform values. (This does not lead to a loss of generality because, as we shall see in the next section, uniform values can be transformed into other distributions.)
Since only finite accuracy is possible on a real computer, we cannot generate continuous random variables. However, it is possible to generate pseudo-random integers, *xk* , uniformly between 0 and some very large number (say *m*), which we use directly as a discrete uniform distribution, or transform into the fraction *xk/m* to be used as an approximation to a uniform continuous distribution between 0 and 1.
The most popular method of generating uniform pseudo-random integers is as follows. First, an initial value *x* 0, the **seed**, is chosen. Thereafter, each number, *xk* , is used to generate the next number in the sequence, *xk* +1, from the relation

$$x_{k+1} := (a * x_k + c) \bmod m$$

The only difficult part is choosing the constants *a*, *c* and *m*. These must be chosen both to ensure that the *period* of the random number generator (i.e., the number of distinct values before the sequence repeats) is large, and also to ensure that consecutive numbers appear to be independent. Note that the period can be at most *m*, because there are only *m* distinct values between 0 and *m*−1.

Progress in a simulation study

1. Problem formulation. Every simulation study begins with a statement of the problem. If the statement is provided by those that have the problem (client), the simulation analyst must take extreme care to insure that the problem is clearly understood. If a problem statement is prepared by the simulation analyst, it is important that the client understand and agree with the formulation. It is suggested that a set of assumptions be prepared by the simulation analyst and agreed to by the client. Even with all of these precautions, it is possible that the problem will need to be reformulated as the simulation study progresses.

2. Setting of objectives and overall project plan. Another way to state this step is "prepare a proposal." This step should be accomplished regardless of location of the analyst and client, viz., as an external or internal consultant.

The objectives indicate the questions that are to be answered by the simulation study. The project plan should include a statement of the various scenarios that will be investigated. The plans for the study should be indicated in terms of time that will be required, personnel that will be used, hardware and software requirements if the client wants to run the model and conduct the analysis, stages in the investigation, output at each stage, cost of the study and billing procedures, if any.

3. Model conceptualization. The real-world system under investigation is abstracted by a conceptual model, a series of mathematical and logical relationships concerning the components and the structure of the system. It is recommended that modeling begin simply and that the model grow until a model of appropriate complexity has been developed. For example, consider the model of a manufacturing and material handling system. The basic model with the arrivals, queues and servers is constructed. Then, add the failures and shift schedules. Next, add the material-handling capabilities. Finally, add the special features. Constructing an unduly complex model will add to the cost of the study and the time for its completion without increasing the quality of the output. Maintaining client involvement will enhance the quality of the resulting model and increase the client's confidence in its use.

4. Data collection. Shortly after the proposal is "accepted" a schedule of data requirements should be submitted to the client. In the best of circumstances, the client has been collecting the kind of data needed in the format required, and can submit these data to the simulation analyst in electronic format. Oftentimes, the client indicates that the required data are indeed available. However, when the data are delivered they are found to be quite different than anticipated. For example, in the simulation of an airline-reservation system, the simulation analyst was told "we have every bit of data that you want over the last five years." When the study commenced, the data delivered were the average "talk time" of the reservationist for each of the years. Individual values were needed, not summary measures. Model building and data collection are shown as contemporaneous in . This is to indicate that the simulation analyst can readily construct the model while the data collection is progressing.

5. Model translation. The conceptual model constructed in Step 3 is coded into a computer recognizable form, an operational model.

6. Verified? Verification concerns the operational model. Is it performing properly? Even with small textbook sized models, it is quite possible that they have verification difficulties. These models are orders of magnitude smaller than real models (say 50 lines of computer code versus 2,000 lines of computer code). It is highly advisable that verification take place as a continuing process. It is ill advised for the simulation analyst to wait until the entire model is complete to begin the verification process. Also, use of an interactive run controller, or debugger, is highly encouraged as an aid to the verification process.

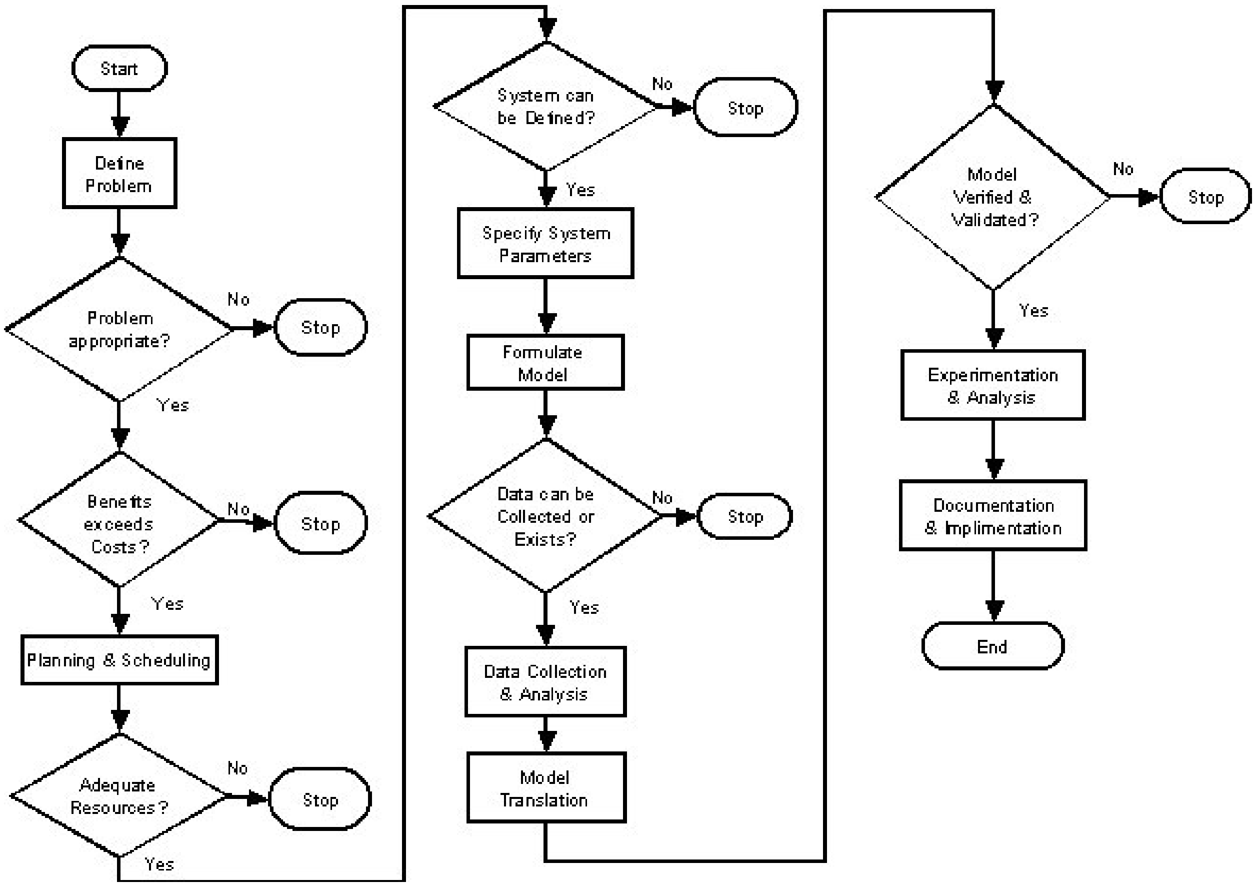
7. Validated? Validation is the determination that the conceptual model is an accurate representation of the real system. Can the model be substituted for the real system for the purposes of experimentation? If there is an existing system, call it the base system, then an ideal way to validate the model is to compare its output to that of the base system. Unfortunately, there is not always a base system. There are many methods for performing validation.
8. Experimental design. For each scenario that is to be simulated, decisions need to be made concerning the length of the simulation run, the number of runs (also called replications), and the manner of initialization, as required.

9. Production runs and analysis. Production runs, and their subsequent analysis, are used to estimate measures of performance for the scenarios that are being simulated.

10. More runs? Based on the analysis of runs that have been completed, the simulation analyst determines if additional runs are needed and if any additional scenarios need to be simulated.

11. Documentation and reporting. Documentation is necessary for numerous reasons. If the simulation model is going to be used again by the same or different analysts, it may be necessary to understand how the simulation model operates. This will enable confidence in the simulation model so that the client can make decisions based on the analysis. Also, if the model is to be modified, this can be greatly facilitated by adequate documentation. The result of all the analysis should be reported clearly and concisely. This will enable the client to review the final formulation, the alternatives that were addressed, the criterion by which the alternative systems were compared, the results of the experiments, and analyst recommendations, if any.

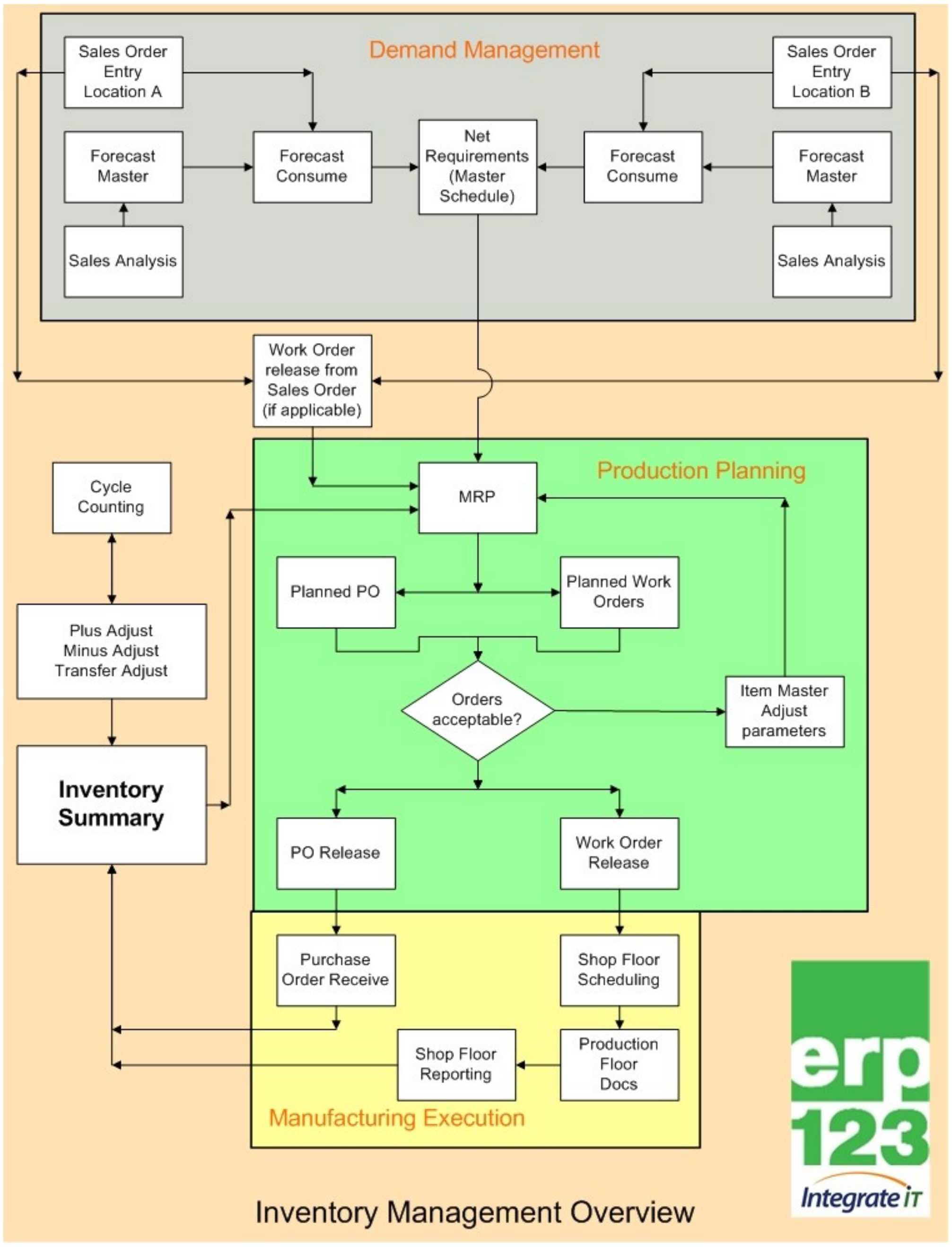
12. Implementation. The simulation analyst acts as a reporter rather than an advocate. The report prepared in step 11 stands on its merits, and is just additional information that the client uses to make a decision. If the client has been involved throughout the study period, and the simulation analyst has followed all of the steps rigorously, then the likelihood of a successful implementation is increased.



Example : An **inventory control system** is a process for managing and locating objects or materials. In common usage, the term may also refer to just the software components.

Modern inventory control systems often rely upon barcodes and radio-frequency identification (RFID) tags to provide automatic identification of inventory objects. Inventory objects could include any kind of physical asset: merchandise, consumables, fixed assets, circulating tools, library books, or capital equipment. To record an inventory transaction, the system uses a barcode scanner or RFID reader to automatically identify the inventory object, and then collects additional information from the operators via fixed terminals (workstations), or mobile computers.

The new trend in inventory management is to label inventory and assets with QR Code, and use smartphones to keep track of inventory count and movement. These new systems are especially useful for field service operations, where an employee needs to record inventory transaction or look up inventory stock in the field, away from the computers and hand-held scanners.



Module-II **(12 Lectures)**
Objective: To use the Probability concepts in Simulation
Events that cannot be predicted precisely are often called random. Many if not most of the inputs to, and processes that occur in systems are to some extent random. Hence, so too are the outputs or predicted impacts, and even people's reactions to those outputs or impacts.
Probability Concepts and Methods

The basic concept in probability theory is that of the random variable. By definition, the value of a random variable cannot be predicted with certainty. It depends, at least in part, on the outcome of a chance event. Examples are: (1) the number of years until the flood stage of a river washes away a small bridge; (2) the number of times during a reservoir's life that the level of the pool will drop below a specified level; (3) the rainfall depth next month; and (4) next year's maximum flow at a gauge site on an unregulated stream. The values of all of these random events or variables are not knowable before the event has occurred. Probability can be used to describe the likelihood that these random variables will equal specific values or be within a given range of specific values.

Distributions of Random Events

Given a set of observations to which a distribution is to be fit, one first selects a distribution function to serve as a model of the distribution of the data. The choice of a distribution may be based on experience with data of that type, some understanding of the mechanisms giving rise to the data, and/or examination of the observations themselves. One can then estimate the parameters of the chosen distribution and determine if the fitted distribution provides an acceptable model of the data.

Stochastic Processes

Historical records of rainfall or stream flow at a particular site are a sequence of observations called a time series. In a time series, the observations are ordered by time, and it is generally the case that the observed value of the random variable at one time influences one's assessment of the distribution of the random variable at later times. This means that the observations are not independent. Time series are conceptualized as being a single observation of a stochastic process, which is a generalization of the concept of a random variable.

Stochastic Variables, Discrete and Continuous Probability function:

The variables that can change with certain probability are called stochastic variables (random variables).

In discrete system simulation we use the probability mass function. If a random variable is a discrete variable, its probability distribution is called a discrete probability distribution.

An example will make this clear. Suppose you flip a coin two times. This simple statistical experiment can have four possible outcomes: HH, HT, TH, and TT. Now, let the random variable X represent the number of Heads that result from this experiment. The random variable X can only take on the values 0, 1, or 2, so it is a discrete random variable.

The probability distribution for this statistical experiment appears below.

Number of heads	Probability
0	0.25
1	0.50
2	0.25

The above table represents a *discrete* probability distribution because it relates each value of a discrete random variable with its probability of occurrence. Binomial and Poisson distribution will also be covered here.

When variable being observed is continuous then an infinite number of possible values can be assumed by the function and we describe the variable by a probability density function. On integrating it within the range we get the cumulative distribution function.

Numerical evaluation of Continuous Probability function and continuous uniformly distributed random number:

The purpose of generating a probability function into a simulation is to generate random numbers with that particular distribution. Customary way of organizing data derived from observations is to display them as a frequency distribution. Relative frequency distribution is a better approach.

By a continuous uniform distribution we mean that the probability of a variable, X, falling in any interval within a certain range of values is proportional to the ratio of the interval size to the range; that is, every point in the range is equally likely to be chosen.

Random number generators: Linear Congruential Generator, Mid Square Method, Multiplicative Congruential Generator, Rejection Method:

Linear congruential generator (LCG) is an algorithm that yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation. The method represents one of the oldest and best-known pseudorandom number generator algorithms. The theory behind them is relatively easy to understand, and they are easily implemented and fast, especially on computer hardware which can provide modulo arithmetic by storage-bit truncation.

The generator is defined by the recurrence relation:

$$X_{n+1} = (aX_n + c) \mod m$$

where X is the sequence of pseudorandom values, and

$m,\ 0 < m$ – the "modulus"
$a,\ 0 < a < m$ – the "multiplier"
$c,\ 0 \le c < m$ – the "increment"
$X_0,\ 0 \le X_0 < m$ – the "seed" or "start value"

are integer constants that specify the generator. If *c* = 0, the generator is often called a multiplicative congruential generator (MCG), or Lehmer RNG. If *c* ≠ 0, the method is called a *mixed congruential generator.*

Mid-square method is a method of generating pseudorandom numbers. In practice it is not a good method, since its period is usually very short and it has some severe weaknesses, such as the output sequence almost always converging to zero. Here we take the mid of a number as the seed value and then square it to generate a random number and accordingly continue till we have not obtained the desired set of random numbers.

The rejection sampling method generates sampling values from an arbitrary probability distribution function $f(x)$ by using an instrumental distribution $g(x)$, under the only restriction that $f(x) < Mg(x)$ where $M > 1$ is an appropriate bound on $f(x)/g(x)$.

Rejection sampling is usually used in cases where the form of $f(x)$ makes sampling difficult. Instead of sampling directly from the distribution $f(x)$, we use an envelope distribution $Mg(x)$ where sampling is easier. These samples from $Mg(x)$ are probabilistically accepted or rejected.

Testing of Random Numbers:

Chi-square is a statistical test commonly used to compare observed data with data we would expect to obtain according to a specific hypothesis. For example, if, according to Mendel's laws, you expected 10 of 20 offspring from a cross to be male and the actual observed number was 8

males, then you might want to know about the "goodness to fit" between the observed and expected. Were the deviations (differences between observed and expected) the result of chance, or were they due to other factors. How much deviation can occur before you, the investigator, must conclude that something other than chance is at work, causing the observed to differ from the expected. The chi-square test is always testing what scientists call the null hypothesis, which states that there is no significant difference between the expected and observed result.
The formula for calculating chi-square (x^2) is:
$x^2= (o-e)^2/e$
That is, chi-square is the sum of the squared difference between observed (*o*) and the expected (*e*) data (or the deviation, *d*), divided by the expected data in all possible categories. Chi-squared distribution, showing X^2 on the x-axis and P-value on the y-axis.

A **chi-squared test**, also referred to as χ^2 **test** (or **chi-square test**), is any statistical hypothesis test in which the sampling distribution of the test statistic is a chi-squared distribution when the null hypothesis is true. Also considered a chi-squared test is a test in which this is *asymptotically* true, meaning that the sampling distribution (if the null hypothesis is true) can be made to approximate a chi-squared distribution as closely as desired by making the sample size large enough. The chi-squared (I) test is used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories. Does the number of individuals or objects that fall in each category differ significantly from the number you would expect? Is this difference between the expected and observed due to sampling variation, or is it a real difference?

Exact chi-squared distribution

One case where the distribution of the test statistic is an exact chi-squared distribution is the test that the variance of a normally distributed population has a given value based on a sample variance. Such a test is uncommon in practice because values of variances to test against are seldom known exactly.

Chi-squared test for variance in a normal population

If a sample of size *n* is taken from a population having a normal distribution, then there is a result which allows a test to be made of whether the variance of the population has a pre-determined value. For example, a manufacturing process might have been in stable condition for a long period, allowing a value for the variance to be determined essentially without error. Suppose that a variant of the process is being tested, giving rise to a small sample of *n* product items whose variation is to be tested. The test statistic *T* in this instance could be set to be the sum of squares about the sample mean, divided by the nominal value for the variance (i.e. the value to be tested as holding). Then *T* has a chi-squared distribution with $n - 1$ degrees of freedom. For example if the sample size is 21, the acceptance region for *T* for a significance level of 5% is the interval 9.59 to 34.17.

Chi-squared test for independence and homogeneity in tables

Suppose a random sample of 650 of the 1 million residents of a city is taken, in which every resident of each of four neighborhoods, A, B, C, and D, is equally likely to be chosen. A null hypothesis says the randomly chosen person's neighborhood of residence is independent of the person's occupational classification, which is either "blue collar", "white collar", or "service". The data are tabulated:

	A	B	C	D		total
Blue collar	90	60	104	95		349
White collar	30	50	51	20		151
Service	30	40	45	35		150
total	150	150	200	150		650

Let us take the sample proportion living in neighborhood A, 150/650, to estimate what proportion of the whole 1 million people live in neighborhood A. Similarly we take 349/650 to estimate what proportion of the 1 million people are blue-collar workers. Then the null hypothesis independence tells us that we should "expect" the number of blue-collar workers in neighborhood A to be

$$\frac{150}{650} \times \frac{349}{650} \times 650 \approx 80.54.$$

Then in that "cell" of the table, we have

$$\frac{(\text{observed} - \text{expected})^2}{\text{expected}} = \frac{(90 - 80.54)^2}{80.54}.$$

The sum of these quantities over all of the cells is the test statistic. Under the null hypothesis, it has approximately a chi-squared distribution whose number of degrees of freedom is

$$(\text{number of rows} - 1)(\text{number of columns} - 1) = (3 - 1)(4 - 1) = 6.$$

If the test statistic is improbably large according to that chi-squared distribution, then one rejects the null hypothesis of **independence**.

A related issue is a test of **homogeneity**. Suppose that instead of giving every resident of each of the four neighborhoods an equal chance of inclusion in the sample, we decide in advance how many residents of each neighborhood to include. Then each resident has the same chance of being chosen as do all residents of the same neighborhood, but residents of different neighborhoods would have different probabilities of being chosen if the four sample sizes are not proportional to the populations of the four neighborhoods. In such a case, we would be testing "homogeneity" rather than "independence". The question is whether the proportions of blue-collar, white-collar, and service workers in the four neighborhoods are the same. However, the test is done in the same way.

The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is designed to test the hypothesis that a given data set could have been drawn from a given distribution. Unlike the chi-square test, it is primarily intended for use with continuous distributions and is independent of arbitrary computational choices such as bin width.

Suppose that we had collected four data points and sorted them into increasing order to get the data set {1.2, 3.1, 5.1, 6.7}. From the pattern of our data alone, we might guess that, if we continued to collect data from this process, 0-25% of our observations would be less than or equal to 1.2, 25-50% would be less than or equal to 3.1, etc.

Perhaps we would like to compare this *empirical* pattern to the pattern we would expect to observe if the data points were drawn from a given theoretical distribution; say, an exponential distribution with a mean of 5 (i.e., l = 1/5 = 0.2). If data points were drawn from this exponential distribution, what fraction would we expect to see below 1.2? Below 3.1? These figures can be computed from the cumulative distribution function for the exponential distribution:

$F(1.2) = 1 - e^{-(0.2)(1.2)} = 0.21$

$F(3.1) = 1 - e^{-(0.2)(3.1)} = 0.46$

$F(5.1) = 1 - e^{-(0.2)(5.1)} = 0.64$

$F(6.7) = 1 - e^{-(0.2)(6.7)} = 0.74$

We compare this to our empirical pattern in Figure 1. The first three rows contain the data points and our highest and lowest estimates of the fraction of the data that would fall below each point. The fourth row contains the result of plugging the data points into the theoretical distribution under consideration (in this case, the exponential distribution with a mean of 5). These values are the *theoretical* estimate of what fraction should fall below each data point. The fifth row is obtained by comparing the fourth row to the second and third rows. Is 0.21 near 0? Near 0.25? We take the absolute value of the larger of the two deviations. For example, in the first column, we get

$|0 = 0.21| = 0.21$

$|0.25 - 0.21| = 0.04$

so, the larger deviation is 0.21. This gives an idea of how far our empirical pattern is from our theoretical pattern.

FIGURE 1: Computing *D* for the Kolmogorov-Smirnov test.

Row 1:	Data point x_1	1.2	3.1	5.1
Row 2:	Empirical fraction falling below data point (low estimate)	0	0.25	0.50
Row 3:	Empirical fraction falling below data point (high estimate)	0.25	0.50	0.75
Row 4:	$F(x_1)$	0.21	0.46	0.64
Row 5:	Largest deviation	0.21	0.21	0.14
Row 6:	Overall largest deviation (*D*)			

Next, we look over the fifth row to find the largest overall deviation (*D*). The largest error, 0.26, is the value of our test statistic. Is this measure of "error" large or small for this situation? To make this judgment, we compare our computed value of this *test statistic* to a critical value from the table in Appendix A. Setting a=0.1 and noting that our sample size is *n*=4, we get a critical value of $D_{4,1.0} = 0.565$. Since our test statistic, *D*=0.26, is less than 0.565, we do not reject the hypothesis that our data set was drawn from the exponential distribution with a mean of 5.

In general, we use the Kolmogorov-Smirnov test to compare a data set to a given theoretical distribution by filling in a table as follows:

• Row 1: Data set sorted into increasing order and denoted as x_i, where i=1,...,n.

• Row 2: *Smallest empirical* estimate of fraction of points falling below x_i, and computed as (i-1)/n for i=1,...,n (e.g. if n=4, this row contains 0, 0.25, 0.50, and 0.75).

• Row 3: *Largest empirical* estimate of fraction of points falling below x_i and computed as i/n for i=1,...,n (e.g., if n=4, this row contains 0.25, 0.50, 0.75, and 1.0).

• Row 4: *Theoretical* estimate of fraction of points falling below x_i and computed as $F(x_i)$, where $F(x)$ is the theoretical distribution function being tested.

• Row 5: Absolute value of difference between row 2 and row 4 or between row 3 and row 4, whichever is larger. This is a measure of "error" for this data point.

• Row 6: The largest "error" from row 5, which gives the test statistic D.

Once this table has been completed, the test statistic D can be compared to the critical value from a statistical table. If the test statistic is larger than the critical value, then we reject the hypothesis that the data set was drawn from the given theoretical distribution; otherwise we do not reject the hypothesis.*

In the preceding example, the parameter of the theoretical distribution (i.e., l = 1/5) was not estimated from the data set. In most cases, *the critical values in the Kolmogorov-Smirnov table are only valid for testing distributions whose parameters have not been estimated from the data*. Had we, for example, used a maximum-likelihood estimate formula to compute l before testing the fit of the distribution, we could not have used the Kolmogorov-Smirnov test. Modified versions of the Kolmogorov-Smirnov test have been developed for testing the fit of a few theoretical distributions in the case where parameter values have been estimated from the data. While we will not cover these modified tests, the ideas behind them are similar to those we have discussed, and many popular statistical software packages perform them.

Generation of Stochastic Variants:

For simulation experiments (including Monte Carlo) it is necessary to generate random numbers (as values of variables). The problem is that the computer is highly deterministic machine - basically, behind each process there is always an algorithm, deterministic computation changing inputs to outputs, therefore it is not easy to generate uniformly spread random numbers over a defined interval or set.

Random number generator is a device capable of producing a sequence of numbers which can not be "easily" identified with deterministic properties. This sequence is then called Sequence of stochastic numbers.

The algorithms typically rely on pseudo random numbers, computer generated numbers mimicking true random numbers, to generate a realization, one possible outcome of a process.

Methods for obtaining random numbers exist for a long time and are used in many different fields (like gaming). However, these numbers suffer from certain bias. Currently the best methods, expected to produce truly random sequences are natural methods that take advantage of the random nature of quantum phenomena.

Arrival Patterns Service Times:

Queuing models

Distribution of request inter-arrival times

The problem: Let us assume that we want to model a system consisting of many users and a server, and we are interested in the response time of the server which is the time between the moment that a user sends a request to the server to the moment when the server has completed its answer (we ignore any communication delays).

Now, let us assume that the service time of the server is a constant (1 second), that is, it always takes exactly 1 second to provide an answer to a request. This means the distribution of the service time is a delta-function with a peak at t = 1 second.

If the users organize themselves in such a way that the next request will be sent to the server only after the last answer was received, then the response time will always be equal to the service time. However, if the users do not coordinate their requests, it may happen that a user sends a request before the last answer was produced. This means that the server is busy when the request arrives and the request has to wait (in some kind of input queue) before it will be processed. Therefore the response time, in this case, will be larger than the service time (service time plus waiting time). We see that the arrival pattern of the request has an impact on the average response time of the system.

Poisson inter-arrival pattern: It can be shown that, if there are very many users without any coordination among them, the inter-arrival time (that is, the time between the arrival of one request up to the arrival of the next request) has an exponential distribution of the form Pt = alpha * exp(-alpha * t) where alpha is the arrival rate. In fact, the average of this inter-arrival distribution is 1/alpha which is the average time between two consecutive requests. We see that in this case there is a good probability that the inter-arrival time is much smaller than the average. Therefore it would be interesting to determine what the average waiting time of requests in the server input queue would be. An answer is given by the queuing models.

Important conclusion: One cannot say what the response time of a server is without knowing the inter-arrival pattern of the requests. The performance guarantee of the server depends on the hypothesis about its environment (which determines the distribution of incoming requests).

Random variate

In the mathematical fields of probability and statistics, a **random variate** is a particular outcome of a random variable: the random variates which are other outcomes of the same random variable might have different values. Random variates are used when simulating processes driven by random influences (stochastic processes). In modern applications, such simulations would derive random variates corresponding to any given probability distribution from computer procedures designed to create random variates corresponding to a uniform distribution, where these procedures would actually provide values chosen from a uniform distribution of pseudorandom numbers.

Procedures to generate random variates corresponding to a given distribution are known as procedures for *random variate generation* or pseudo-random number sampling.

In probability theory, a random variable is a measurable function from a probability space to a measurable space of values that the variable can take on. In that context, and in statistics, those values are known as a **random variates**, or occasionally **random deviates**, and this represents a wider meaning than just that associated with pseudorandom numbers.

Definition

Devroye defines a random variate generation algorithm (for real numbers) as follows:

Assume that

1. Computers can manipulate real numbers.
2. Computers have access to a source of random variates that are uniformly distributed on the closed interval $[0; 1]$.

Then a random variate generation algorithm is any program that halts almost surely and exits with a real number *X*. This *X* is called a **random variate**.

(Both assumptions are violated in most real computers. Computers necessarily lack the ability to manipulate real numbers, typically using floating point representations instead. Most computers lack a source of true randomness (like certain hardware random number generators), and instead use pseudorandom number sequences.)
The distinction between *random variable* and *random variate* is subtle and is not always made in the literature. It is useful when one wants to distinguish between a random variable itself with an associated probability distribution on the one hand, and random draws from that probability distribution on the other, in particular when those draws are ultimately derived by floating-point arithmetic from a pseudo-random sequence.

Random Variate Generation

It is assumed that a distribution is completely specified and we wish to generate samples from this distribution as input to a simulation model.

Techniques

- Inverse Transformation
- Acceptance-Rejection
- Convolution

All these techniques assume that a source of uniform (0, 1) random numbers is available; R_1, R_2,..., where each R_i has:

ì1 ,   0 £ x £ 1

pdf: $f_R(x)$ =   í

î0 ,   otherwise   and

ì0 ,   x < 0

cdf: $F_R(x)$ =   íx ,   0 £ x £ 1

î1 ,   x > 1

Note: The random variable may be either discrete or continuous.
If the random variable is discrete, ==>
x take on a specific value, and F(x) is a step F^n
If F(x) is continuous over the domain x, ==>
f(x) = dF(x) / dx   and
the derivative f(x) is called the pdf.
Mathematically, the cdf is:
$F(x) = P(X \leq x) =$, where F(x) is defined over the range $0 \leq F(x) \leq 1$, and f(t) represents the value of the pdf of the variable x, when X = t.

Example #1
Generate random variates x with density function f(x) = 2x, 0 £ x £ 1

Solution:

$F(x) =$ $= x^2$, 0 £ x £ 1

Now set $F(x) = R$ ==> $R = x^2$

Next, solve for x, ==> $x = F^{-1}(R) =$ ÖR, 0 £ r £ 1

\Values of x with pdf $f(x) = 2x$ can be generated by taking the square root of the random, R.

Example #2

Generate random variates x with density function

$f(x) = \lambda e^{-\lambda x}$, $0 \le x$ f(x) = í

0 , $x < 0$

Solution:

$F(x) =$

$= 1 - e^{-\lambda x}$, $0 \le x$

$= 0$ $x < 0$Now set $F(x) = R$

Next solve for x, ==>

$1 - e^{-\lambda x} = R$

$e^{-\lambda x} = 1 - R$

$-\lambda x = \ln(1 - R)$

$x = -\{\ln(1 - R)\} / \lambda$

or $= -\{\ln(R)\} / \lambda$

- Uniform Distribution

  Consider a random variable X that is uniformly distributed on the interval [a, b]

  pdf: $f(x) = 1/(b-a)$ $a \le x \le b$

  0 , otherwise

  To generate random variates:

  Step 1. $F(x) =$ 0 $x< a$

  $(x - a) / (b - a)$, $a \le x \le b$

  1 $x>b$

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

  Step 3. $X = a + (b - a) R$

Module III
Objective: Know about simulation Language like GPSS:

1. Discrete events

A model used in discrete system simulation has a number of numbers to represent the state of a system. A number used to represent some aspect of the system state is called a state descriptor. Some state descriptors range over values that have physical significance, such as the number representing the count of documents .As the simulation proceeds ,the state descriptors change value. A discrete event is a set of circumstances, that causes an instantaneous change in one or more system state descriptors. It is possible that two different events occur simultaneously, or are modeled as being simultaneous, so that not all changes of state descriptors occurring simultaneously necessarily belong to a single event.

2. Representation of time

The passage of time is recorded by a number referred to as a clock time.It is usually set to zero at the beginning of a simulation and subsequently indicates how many units of simulated time have passed since the beginning of the simulation. Here the simulation time means the clock time. Ratio of simulated time to the real time taken can vary enormously.
Two basic methods exist for updating clock time. One method is to advance the clock to the time at which the next event is due to occur. The other method is to advance the clock by small intervals of time and determine at each interval whether an event is due to occur at that time. Here the first method is called event oriented and the second method is called interval oriented. Discrete system simulation is usually carried out with the use of event oriented method whereas continuous system simulation uses interval oriented method. There is no firm rule to represent time in simulation

3. Generation of arrival pattern

It is a important aspect of discrete system simulation. It is possible that an exact sequence of arrivals has been specified for the simulation. Also the sequence of inputs may have been generated from observations on a system. Computer system designs and especially the programming components of the system are often tested with a record, gathered from a running system that is representative of the sequence of operations the computer system will have to execute. This approach is called trace driven simulation.
When there is no interaction between the exogenous arrivals and the endogenous events of the system,a sequence of arrivals in preparation for the simulation is created. Usually simulation proceeds by creating new arrivals as they are needed.
The exogenous arrival of an entity is defined as an event and the arrival time of the next entity is recorded as one of the event times. When the clock time reaches this event time, the event of entering the entity is calculated at once from the inter arrival time distribution.

4. Simulation of a telephone System

A telephony system has a number of telephones, connected to a switchboard by lines. Here the object of the simulation will be to process a given number of calls and determine what proportion are successfully completed, blocked, or found to be busy calls.

5. Delayed calls

Suppose we assume that calls that cannot be connected are not lost. Instead, they wait until they can be connected. This case does not happen in a normal telephony system involving human beings talking to each other. But it can happen to messages in a switching system that has store and forward capability.

6. Priorities and parameters

A transaction has no particular identity. Each is treated by a block in the same manner as any other transaction. Each transaction has one of 128 levels of priority, indicated by the numbers 0 to 127,with 0 being the lowest priority. At any point in the block diagram, the priority can be set up or down to any of the levels by the PRIORITY block. The block is coded by putting the priority in field A of the block.

It is also possible to designate the priority at the time a transaction is created by putting the priority in the E field of the GENERATE block creating the transaction. If the field is left blank, the priority is set to 0.

A transaction has also parameters, which carry numerical data that can affect the way the transaction is processed by a block. The values are identified by the notation Pxn, where n is the parameter number, and x is the type. If no declaration is made, the transaction have 12 half word parameters.

All parameter values are zero at the time a transaction is created. A value is given to a parameter when a transaction enters an ASSIGN block. The number of the parameter is given in the field A of the ASSIGN block. The value of the parameter to be taken is given in the field B.

7. Standard Numerical Attributes

Every entity in a system has some attributes, for e.g. number of transactions in a storage or the length of a queue, which are made available to the program user. These attributes, collectively are called standard numerical attributes. Each type of SNA is identified by a one or two letter code and a number.

Many uses can be made of SNA's. They provide the inputs to functions, and hence allow a great variety of functional relationships to be introduced into the model. Values of the SNA's change as the simulation proceeds. The program does not continuously maintain current value; it completes value of SNA's at the time they are needed. Some example of GPSS SNA are C1, CHn, Fn, Kn, M1 etc.

8. General description of GPSS

The system to be simulated in GPSS is described as a block diagram in which the blocks represents the activities, and lines joining the blocks indicates the sequences in which the activities can be executed.

The approach taken in GPSS is to define a set of 48 specific block types, each of which represents a characteristic action of systems. The program user must draw a block diagram of the system using only these block types.

Moving through the system being simulated that depends upon the nature of system. The sequence of events in real time is reflected in movement of transaction s from block to block in Fimulated time.

Transaction are created at one more GENERATE blocks and are removed from the simulation at TERMINATE blocks . There can be many transactions simultaneously moving through the block diagram. Each transaction is always positioned at a block and most blocks can hold many transactions simultaneously. The transfer of transaction from one block to another occurs instantaneously at a specific time or when some change of system condition occurs.

9. Facilities and Storages

A facility is defined as an entity that can be engaged by a single transaction at a time. A storage is defined as an entity that can be occupied by many transaction at a time, up to some predetermined limits. A transaction controlling a facility, however can be interrupted or preempted by another transaction. In addition, both facilities and storages can be made unavailable again, as occurs when a repairs has made.

There can be many instances of each type of entity to a limit set by the program (usually 300). Individual entities are identified by number, a separate number sequence being used for each type.

Four block types, SEIZE, RELEASE, ENTER, and LEAVE, concerned with using facilities and storages. Field A in each case indicates which facility and storage intended, and choice is usually marked in the flag attached to the system of blocks. The SEIZE block allows a transaction to engage a facility if it is available. The RELEASE block allows the transaction to disengage the facility. In an analogous manner, an ENTER block allows a transaction to occupy space in storage, if it is available, and LEAVE block allows it to give up the space.

10. Gathering Statistics

Certain block types in GPSS are constructed for the purpose of gathering statistics about the system performance, rather then of representing system actions. The QUEUE, DEPART, MARK, and TABULATE blocks shown in fig 9-1 serve this purpose. They introduce two other entities of GPSS program, queues and tables.

The QUEUE block increases and change is a unit change ; otherwise the value of field B (>=1) is used. The program measured the average and maximum queue lengths and, if required, the distribution of time spent on queue.

It is also described to measure the length of time taken by transaction to move through the system or parts of the system, and this can be done with MARK and TABULATE blocks. Each of these blocks types notes the time a transaction arrives at the block. The MARK block simply notes the time of arrival on the transaction. The TABULATE block subtracts the time noted by a MARK block from the time of arrival at the TABULATE block.

Module-IV

Objective: To focus on the simulation languages, analysis and testing of simulation outputs and recent trends .

Simulation languages:

A simulation programming language **(SPL)** is a POL(Problem oriented language) with special features. Simulation being a problem-solving activity with its own needs, programming languages have been written to rake special features available. **General Purpose Simulation System** (GPSS) (originally **Gordon's Programmable Simulation System** after creator Geoffrey Gordon; the name was changed when it was decided to release it as a product) is a discrete time simulation general-purpose programming language, where a simulation clock advances in discrete steps. A system is modelled as transactions enter the system and are passed from one service (represented by blocs) to another. This is particularly well-suited for problems such as a factory. GPSS is less flexible than simulation languages such as Simula and SIMSCRIPT II.5 but it is easier to use and more popular.

The GPSS world view

The GPSS code may surprise many programmers with experience in procedural, object oriented or functional programming. The world is rather simulated with entities moving through the model. These entities, called *Transactions*, are envisioned as moving from *Block* to *Block*, where a Block is a line of code and represents unit actions that affects the Transaction itself or other entities.

These other entities can be broadly classified in Resources, Computational entities and Statistical entities.Resources, like *Facilities* and *Storages* represent limited capacity resources. Computational entities, like *Ampervariables* (variables), *Functions* and random generators are used to represent the state of Transactions or elements of their environment. Statistical entities, like *Queues* or *Tables* (histograms) collect statistical information of interest.

Sample code

The following example, taken from *Simulation using GPSS*, is the "Hello world!" of GPSS and will illustrate the main concepts.

The aim is to simulate one day of operation of a barber shop. Customers arrive in a random constant flow, enter the shop, queue if the barber is busy, get their hair cut on a first-come first-served basis, and then leave the shop. We wish to know the average and maximum waiting line, as well as the number of customers.

```
      SIMULATE              ; Define model
*
*  Model segment 1
*
      GENERATE 18,6         ; Customer arrive every 18±6 mn
      QUEUE    Chairs        ; Enter the line
      SEIZE    Joe           ; Capture the barber
      DEPART   Chairs        ; Leave the line
      ADVANCE  16,4          ; Get a hair cut in 16±4 mn
      RELEASE  Joe           ; Free the barber
      TERMINATE              ; Leave the shop
*
*  Model segment 2
*
      GENERATE 480           ; Timer arrives at time = 480 mn
      TERMINATE 1            ; Shut off the run
*
*  Control cards
*
      START     1            ; Start one run
      END                    ; End model
```

The "program" is comprised between the `SIMULATE` and `END` statements, and is divided into "model segments" and "control cards".

The first segment models customers. The `GENERATE` block creates a flow of Transactions and schedules them to enter the model with an inter-arrival time uniformly distributed over the range 18±6. It is the programmer's responsibility to interpret these transaction as customers and to understand that the time is to be counted in minutes. The Transactions start their existence in the `GENERATE` block and progress from Block to Block, according to certain rules, until they reach a `TERMINATE` which remove them from the model.

Normally transactions progress from one block to the next one, so the customer transactions will leave the `GENERATE` block to enter the `QUEUE Chairs` block. This block simulates a waiting line, and collects statistics accordingly. In the example, it materialize a line of chairs and, at the end of the simulation, we will know, among other things, the maximum queue size (how many chairs are needed) and the average waiting time. The `QUEUE` block requires the name of the queue as a parameter, because more than one queue may exist in the model. Each one is associated with a `DEPART` block, which is triggered when the transaction leaves the queue. GPSS remembers which transactions are in the queue, so that it possible to know the average time spent, and to check that no buggy transaction is leaving a queue without previously entering in it.

After the `QUEUE chairs` block, the transaction will try to proceed to the `SEIZE Joe` block, a block simulating the capture of the *Facility* named Joe. Facilities model single servers of capacity one. If the facility is busy, the `SEIZE` will deny the attempting transaction the right to enter. In the example, the customer will wait in the `QUEUE` block. If it is free, or as soon as it becomes available, the transaction will be allowed to capture the facility, mark it as busy to others transactions and start to count the service time and other statistics, until the same transaction passes the corresponding `RELEASE Joe` block.

The `SEIZE` / `RELEASE` pairs are linked by the facility name, because many independent facilities may exist in the model. They can model operators, like a barber, a repairman, an agent, but also pieces of equipment, like a crane, a gas station, an authorization document, etc., in fact anything with capacity one. To simulate multiple parallel servers, like a team of five barbers, or an oven with a capacity of 10, GPSS uses entities named `STORAGE`s.

After a customer seizes Joe, she proceeds to the next statement which is `ADVANCE 16,4`, whose task is to freeze the entity for a prescribed length of time, here a random number picked between 16-4=12 and 16+4=20mn. Other service time distributions are available through GPSS `FUNCTION` (a somehow different notion than function in other programming languages). During that time, other transactions will be allowed to move through the model, blocking some other facilities that may exist in the model, but not Joe because this facility is busy with the frozen customer. After the prescribed time, the customer will wake up, proceed to the next statement, which will free Joe, and `TERMINATE`.

Then the next transaction on the previous block, that is a customer sitting on a chair, will be able to `SEIZE Joe`. To select the "next" transaction, GPSS uses the first-come first-served basis, with priority. Other selection policies can be programmed by direct manipulation of the *future event chain* entity.

In parallel to this first segment, simulating the customer behavior, a second model segment simulates the end of the day. At time 480mn = 8h a entity is `GENERATE`d, which will `TERMINATE` on the next block. This time, the `TERMINATE` as a parameter of 1, meaning a special counter is decreased by 1. When that counter reaches 0, the program stops and the output is printed. This special counter is setup with the `START` statement. In the example, it is set to one, thus the simulation will finish after one run of 480 mn in simulated time.

It indicates that Joe was busy 86.0% of the time, gave a hair cut to 26 customers and that hair cut took 15.88 minutes on the average. Incidentally, Joe was cutting the hair of customer number 26 when the simulation was closed. No programming provisions were taken for the barber to finish the hair cut before to close the shop.

It indicates also that a maximum of 1 customer was observed waiting his turn, in facts the number of waiting customer was on the average 0.160. A total of 27 customers did enter the queue, so that customer number 27 was still sitting, waiting his turn, when Joe closed the shop. Out of these 27 customers, 12 were served without having to wait. In facts, the queue was empty 44.4% of the time. The average waiting time was 2.851 mn, and the average waiting time for the 15=27-12 customers who did really wait was 5.133 m

Output Analysis in Stochastic Simulation

Once the simulation program has been completely written and carefully tested, the actual simulation experiment must be carried out. The idea of the experiment is to conduct one or more simulation ‘‘runs’’ from which measurements are collected. The output from these runs is used to assess the performance of the system (or sys-tems, if alternative strategies are being evaluated). For example, one run from our service station simulator might have produced the following output:

```
This
simulat
ion run
uses
three
pumps
and the
followi
ng
random
number
seeds:
      1     2       3       4
Curre                       Averag        Averag
nt    Total NoQueue Car->Care      Numbere      Pump  Total  Lost
```

Time	Cars	Fraction	Time	Litres	Balked	Wait	Usage	Profit	Profit
20000	382	0.529	52.356	34.342	67	54.877	0.875	221.34	46.62
40000	771	0.470	51.881	34.454	136	59.949	0.887	505.85	98.25
60000	1157	0.475	51.858	34.767	206	61.015	0.888	794.76	150.87
80000	1554	0.476	51.480	35.052	290	60.764	0.884	1088.26	213.51
100000	1967	0.475	50.839	34.974	379	60.524	0.884	1376.51	283.35
120000	2351	0.477	51.042	34.957	451	60.753	0.882	1652.25	342.38
140000	2760	0.473	50.725	34.824	539	61.889	0.883	1934.74	408.09
160000	3175	0.468	50.394	34.682	625	62.332	0.886	2225.05	467.86
180000	3574	0.464	50.364	34.753	696	62.592	0.888	2524.16	521.05
200000	3957	0.468	50.543	34.746	769	61.654	0.887	2807.68	569.61

2.9.1. Initial and Final Conditions

When calculating performance figures from a simulation run, care must be taken to ensure that the results obtained are not systematically distorted because of either the initial state of the system at the beginning of each simulation run or the final state of the system when end-of-simulation is encountered.

For example, suppose we were concerned with the time for an auto to fill up its tank during a busy time of day, such as afternoon rush hour before a holiday. If we ran the simulation for a long time, we might find that under those operating conditions an average of 15 autos are in the station having their tanks filled or waiting for a free pump. But if the initial state for the simulation did not have any autos in the service station, then it would be obvi-ous that the first few autos spent less time waiting than a ‘‘typical’’ auto would. Thus, our estimate for the mean system time would turn out too low unless we

i) ran our simulation for such a long time that the measurements from these first few customers had very little influence on the mean,

ii) began our measurements part way through the simulation run, by which point the system should have reached a state that is more ‘‘normal’’, or

iii) used the average values from one simulation run as the initial conditions for the next simulation run so that, hopefully, the system is in a normal state throughout the entire run.

In the output from our service station simulator, the effect of the initial conditions (i.e., a completely idle sta-tion) are quite evident. Notice that even though the first progress report was not printed until almost 400 cars had passed through our service station, we still see a downward trend in the fraction of time that the queue of waiting cars is empty, and an upward trend in both the average waiting time and the utilization of the pumps.

The end conditions of the simulation must also be checked before we can trust our calculated performance statistics. For example, the waiting times and service times for customers in the system when end-of-simulation is reached may or may not be counted, depending on the method used for recording these measurements. If such cus-tomers are not counted, then we risk underestimating the utilization of the server, because some customers who might have been part way through their service time are ignored. If such customers are counted, then we risk underestimating the mean waiting time and overestimating the mean service rate because, in effect, we may be claiming that these customers were served during our simulation run when in fact their service would not have been completed until later.

In the case of our service station simulator, notice that we are adding the service times for each car to the total service performed in the startService procedure. Thus, we are always overestimating the pump utilization: in effect, we are (optimistically) claiming that the entire service times for all cars not still in the waiting queue have been completed by the end of simulation — even those cars still in service. If the run length is short enough, this choice of stopping condition can lead to ‘‘impossibly good’’ results. For example, notice that in the following data (representing a short run with a heavily loaded service station) the pumps appear to be busy more than 100 percent of the time at the first few progress reports!

```
This
simulat
ion run
uses
two
pumps
and the
followi
ng
random
number
seeds:
      1     2       3       4
Curre Tota          Car->Ca Averag        Averag
nt    l     NoQueue r       e      Number e      Pump  Total  Lost
! Time___ _ _Cars__ _ _  Fraction_____ _ _ _Time__ _ _ _  Litres____ _ _Balked____ _
_Wait__ _ _ _Usage___ _  Profit____ _ _Profit___ _ _ _ _ _ _ _ _ _ _
                                          182.89
2000  39    0.072   51.282  31.508 11     7      1.002 -18.43 9.15
                                          184.42
4000  82    0.039   48.780  31.037 28     9      1.004 2.70   20.93
                                          190.62
6000  124   0.059   48.387  33.032 46     0      1.014 26.29  36.10
                                          190.95
8000  162   0.068   49.383  34.455 62     9      1.005 50.97  48.57
                                          205.27
10000 207   0.069   48.309  33.815 81     7      1.002 71.16  63.84
                                          204.17
12000 240   0.067   50.000  34.090 92     6      1.005 92.20  72.33
                                          215.16
14000 284   0.058   49.296  34.094 111    1      1.005 115.38 86.68
                                          210.37
16000 304   0.088   52.632  34.567 113    7      0.993 134.53 88.18
                                          210.08
18000 343   0.094   52.478  34.308 127    6      0.996 155.13 99.06
                                          211.13
20000 382   0.084   52.356  34.227 140    8      0.997 178.59 108.28
```

Of course, we could easily replace the overestimate with an underestimate by moving the accumulation of service time to the departure routine. However, calculating the correct value for pump utilization is much trickier.

Another possibility that must be considered in assessing the end conditions of the simulation is whether or not the system being simulated is overloaded. Since only a finite period of simulated time elapses during each simula-tion run, the average waiting time will always be finite, even if the system is in fact so overloaded that the customer waiting times are growing without bound as simulated time advances. Thus, our performance statistics are unreli-able if a significant proportion of the customers are still in the system at the end of simulation: either the simulation run was too short (and the effects of the initial and final conditions are large), or the system was overload.

Confidence Intervals on the Mean of a Random Variable

Once we have carried out one or more simulation runs, and convinced ourselves that the boundary conditions due to initial and final conditions have not spoiled our results, we are left with the task of trying to conclude some-thing about the values of various random variables in the model, such as the mean waiting time for an auto in our service station, for example. It is simple to calculate the **sample mean**

i.e., the average of the N measurements that we made of the value of the random variable X. Similarly, we can calculate the **sample variance:**

by averaging the squared differences between each measurement and the sample mean. But as we scan down the columns of measurements in the output, it is obvious that these sample quantities vary as a function of (simulated) time. Furthermore, if we repeat the experiment using different random number seeds, we soon see that the measure-ments also vary from one run to the next. Consequently, we should be skeptical of concluding that any of these sample means or variances are equal to the true mean and variance of X that we set out to find in the first place. And indeed to proceed further, we need to make some assumptions about the distribution of X.

As a first step, suppose a ‘‘helpful genie’’ were to tell us that the random variable X has a **Normal distribution** (i.e., a ‘‘bell-shaped’’ probability density function). In this case, knowledge of the Normal distribution would allow us to construct **confidence intervals** about the sample mean, say X* ± C, that contain the true mean, say M, a specified fraction of the time (the greater the fraction, the wider the resulting confidence interval about X+). This construction is based on the following two facts about the Normal distribution with mean M and variance

σ^2 . First, the proportion of measurements that fall within a distance of $K\sigma$ from M is known. For example, if we make a large number of measurements x_1 , x_2 , . . . , then approximately 95 percent of them will fall within the interval $M \pm 2\sigma$. The key observation for us is that, since the distance from M to a measurement x_j is the same as the distance from x_j to M, we can turn this result around to say that 95 percent of the time M falls within the interval $x_j \pm 2\sigma$. A table of results in this form is shown below in Figure 5. Second, the average of N measurements taken from a Normal distribution with mean M and variance σ^2 also has a Normal distribution, but with mean M and variance σ^2 /N. Thus, if the confidence interval ends up too wide, it can be tightened by averaging in additional measurements. For example, using four times as many measurements cuts the width of the confidence interval in half.

Figure 5. Intervals about a measurement x that will contain the true mean a given fraction of the time, when the ran-dom variable has a Normal distribution with mean M and variance σ^2

The problem with the result above is that even if we knew that the random variable had a Normal distribution, we still couldn't write down a confidence interval for M without knowing σ^2 (which depends on M). To get out of this circular chain of reasoning, we need a method of estimating σ^2 that is based on X**0** instead of M. Clearly the sample variance, defined above, is not a very good estimate, especially when the number of measurements is small. Indeed, if only one measurement is available, then the sample variance is always zero, indicating that we have not observed any variability in our measurements. But $\sigma^2 = 0$ means something quite different, namely that X is not a random variable at all! The solution is to use an ''unbiased'' estimate of the sample variance to represent σ^2 , i.e.,

$$\sigma^2 \sim \frac{1}{N-1}\sum_{j=1}^{N}\left(x_j - \bar{X}\right)^2 ,$$

and then simply go ahead and use the results described above in Table 1. (By dividing the sum by $N - 1$ instead of N, it is clear that we cannot estimate σ^2 from a single measurement.)

A more serious problem with the method outlined above is that the kinds of random variables that usually come up in simulation models, such as the mean time in system for autos at our service station, almost never have a Normal distribution! Fortunately, the **Central Limit Theorem** in statistics tells us that under fairly general cir-cumstances, the average of a large number of measurements does have a Normal distribution even if the individual measurements themselves do not have a Normal distribution. It follows that we can still find a confidence interval if we carry out K independent repetitions of the same simulation experiment and treat the K sample means X**7**$_j$, $j = 1, . . . , K$, as the measurements.

For example, suppose we ran four more runs of the service station simulation with 3 pumps, giving us, in total, five independent measurements of the mean waiting time. The last line of output from each of these runs is reproduced in the following table:

```
Current                                                         Total
NoQueue    Car->Car    Average    Number    Average    Pump Total
       Lost
```

```
8_Time8_8_8_8_8_8Cars_8_8_8_8Fraction_8_8_8_8_8_8_8_Time8_8_8_8_8_8Lit
res_8_8_8_8_8_Balked8_8_8_8_8_8_Wait8_8_8_8_8_Usage8_8_8_8_8Profit_8_8
_8_8_8_Profit8_8_8_8_8_8_8_8_8_8_8_8_
20000                                                      2807.6 569.6
0     3957  0.468  50.543  34.746  769   61.654  0.887 8      1
20000                                                      2870.2 662.0
0     4096  0.397  48.828  35.081  881   76.382  0.903 0      9
20000                                                      2826.6 652.2
0     4062  0.432  49.237  34.850  848   66.428  0.899 9      9
20000                                                      2821.6 594.9
0     3961  0.440  50.492  35.108  787   69.013  0.894 6      3
20000                                                      2799.5 604.6
0     3972  0.429  50.352  34.886  790   66.098  0.890 3      2
```

Variance Reduction Techniques

In a simulation study, we are interested in one or more performance measures for some stochastic model.Methods that one can use to reduce the variance of the estimator W . We will successively describe the following techniques:

Common Random Numbers

Antithetic Variables

Control Variates

Conditioning

The first method is typically used in a simulation study in which we want to compare performance measures of two different systems. All other methods are also useful in the case that we want to simulate a performance measure of only a single system.

Common Random Numbers

The idea of the method of common random numbers is that if we compare two different systems with some random components it is in general better to evaluate both systems with the same realizations of the random components. Key idea in the method is that if X and Y are two random variables, then

$$\text{Var.X} - \text{Y/} \; D \; \text{Var.X/} \; C \; \text{Var.Y/} - 2\text{Cov.X}; \; \text{Y/}:$$

Hence, in the case that X and Y are *positively* correlated, i.e. $\text{Cov.X}; \text{Y/} > 0$, the variance of $X - Y$ will be smaller than in the case that X and Y are independent. In general, the use of common random numbers leads to positive correlation of the outcomes of a simulation of two systems. As a consequence, it is better to use common random numbers instead of independent random numbers when we compare two different systems.

Let us illustrate the method using the following scheduling example. Suppose that a finite number of N jobs has to be processed on two identical machines. The processing times of the jobs are random variables with some common distribution function F . We want to compare the completion time of the last job, C_{max}, under two different policies. In the LPTF policy, we always choose the remaining job with the longest processing time first, in the SPTF policy we choose the remaining job with the shortest processing time first.

In Table 5 and Table 6 we compare for $N = 10$ and $F(x) = 1 - e^{-x}$ the estimators and confidence intervals for $E(C_{\max}^{\text{SPTF}} - C_{\max}^{\text{LPTF}})$ when we do not, resp. do, use common random numbers.

We conclude that in this example the use of common random numbers reduces the standard deviation of the estimator and hence also the length of the confidence interval with a factor 5.

30

# of runs	mean	st. dev.	95% conf. int.
1000	0.8138	2.5745	[0.645, 0.973]
10000	0.8293	2.4976	[0.780, 0.878]
100000	0.8487	2.4990	[0.833, 0.864]
1000000	0.8398	2.4951	[0.835, 0.845]

Table 5: Estimation of $E(C_{\max}^{\text{SPTF}} - C_{\max}^{\text{LPTF}})$ without using common random numbers

# of runs	mean	st. dev.	95% conf. int.
1000	0.8559	0.5416	[0.822, 0.889]
10000	0.8415	0.5230	[0.831, 0.852]
100000	0.8394	0.5164	[0.836, 0.843]
1000000	0.8391	0.5168	[0.838, 0.840]

Table 6: Estimation of $E(C_{\max}^{\text{SPTF}} - C_{\max}^{\text{LPTF}})$ using common random numbers

When we want to use common random numbers, the problem of synchronization can arise: How can we achieve that the same random numbers are used for the generation of the same random variables in the two systems?

In the previous example, this synchronization problem did not arise. However, to illustrate this problem, consider the following situation. In a $G/G/1$ queueing system the server can work at two different speeds, v_1 and v_2. Aim of the simulation is to obtain an estimator for the difference of the waiting times in the two situations. We want to use the same realizations of the interarrival times and the sizes of the service requests in both systems (the service time is then given by the sizes of the service request divided by the speed of the server). If we use the program of the discrete event simulation of Section 3 of the $G/G/1$ queue, then we get the synchronization problem because the order in which departure and arrival events take place depends on the speed of the server. Hence, also the order in which interarrival times and sizes of service requests are generated depend on the speed of the server.

The synchronization problem can be solved by one of the following two approaches:

- Use separate random number streams for the different sequences of random variables needed in the simulation.

- Assure that the random variables are generated in exactly the same order in the two systems.

For the example of the $G/G/1$ queue, the first approach can be realized by using a separate random number stream for the interarrival times and for the service requests. The second approach can be realized by generating the service request of a customer already at the arrival instant of the customer.

7.2 Antithetic Variables

The method of antithetic variables makes use of the fact that if U is uniformly distributed on $.0; 1/$ then so is $1 - U$ and furthermore U and $1 - U$ are negatively correlated. The key idea is that, if

W_1 and W_2 are the outcomes of two successive simulation runs, then

$$\mathrm{Var}\left(\frac{W_1+W_2}{2}\right)=\frac{1}{4}\mathrm{Var}(W_1)+\frac{1}{4}\mathrm{Var}(W_2)+\frac{1}{2}\mathrm{Cov}(W_1,W_2).$$

Hence, in the case that W_1 and W_2 are *negatively* correlated the variance of $(W_1+W_2)/2$ will be smaller than in the case that W_1 and W_2 are independent.

The question remains how we can achieve that the outcome of two successive simulation runs will be negatively correlated. From the fact that U and $1-U$ are negatively correlated, we may expect that, if we use the random variables $U_1,\ldots,U_m$ to compute W_1, the outcome of the first simulation run, and after that $1-U_1,\ldots,1-U_m$ to compute W_2, the outcome of the second simulation run, then also W_1 and W_2 are negatively correlated. Intuition here is that, e.g., in the simulation of the $G/G/1$ queue large realizations of the U_i 's corresponding to large service times lead to large waiting times in the first run. Using the antithetic variables, this gives small realizations of the $1-U_i$'s corresponding to small service times and hence leading to small waiting times in the second run.

We illustrate the method of antithetic variables using the scheduling example of the previous subsection.

# of runs	mean	st. dev.	95% conf. int.
1000	5.0457	1.6201	[4.945, 5.146]
10000	5.0400	1.6020	[5.009, 5.071]
100000	5.0487	1.5997	[5.039, 5.059]
1000000	5.0559	1.5980	[5.053, 5.059]

Table 7: Estimation of $E(C_{\max}^{\mathrm{LPTF}})$ without using antithetic variables

# of pairs	mean	st. dev.	95% conf. int.
500	5.0711	0.7216	[5.008, 5.134]
5000	5.0497	0.6916	[5.030, 5.069]
50000	5.0546	0.6858	[5.049, 5.061]
500000	5.0546	0.6844	[5.053, 5.056]

Table 8: Estimation of $E(C_{\max}^{\mathrm{LPTF}})$ using antithetic variables

In Table 7 and Table 8 we compare, again for $N=10$ and $F(x)=1-e^{-x}$, the estimators and confidence intervals for $E(C_{\max}^{\mathrm{LPTF}})$ when we do not, resp. do, use antithetic variables. So, for example, we compare the results for 1000 independent runs with the results for 1000 runs consisting of 500 pairs of 2 runs where the second run of each pair uses antithetic variables. We conclude that in this example the use of antithetic variables reduces the length of the confidence interval with a factor 1.5.

Finally, remark that, like in the method of common random numbers, the synchronization problem can arise. Furthermore, it should be noted that the method is easier to implement if all random variables are generated using the inversion transform technique (only one uniform random number is needed for the realization of one random variable) than if we use, e.g., the rejection method to generate random variables (a random number of uniform random numbers are needed for the realization of one random number).

Control Variates

The method of control variates is based on the following idea. Suppose that we want to estimate some unknown performance measure $E(X)$ by doing K independent simulation runs, the i-th one yielding the output random variable X_i with $E(X_i) = E(X)$. An unbiased estimator for $E(X)$ is $\sum_{i=1}^{K} X_i / K$. However, assume that at the same time we are able to

E.X/ is given by N

a related output variable *Y* *E.Y/* D *E.Y/* *E.Y/*

is known. If we denote by

simulate *K* P , with *i* *I* , where

C

/=K , then, for any constant *c*, the quantity

N D *i* D1 *Yi* N N – is also an unbiased

estimate of *E.X/*. Furthermore, from the formula

P C C *c* 2

2 Cov N N *c* , where

Var N D Var . N

N – Var N

C

it is easy to deduce that Var. X C c.Y E .Y /// is minimized if we take c D

re

N

N –

x

Cov. N N

c Y :

D –

Var. N

are usually not known beforehand and must

x ; Y .Y /

Unfortunately, the quantities Cov. N and Var N

be N Y is control variate fo

estimated from the simulated data. The quantity

called a

N

are positively correlated. If a simulation

^{x}N N N

$_x$ $_Y$

. and

To see why the met

hodworks, suppose that

is larger than

Y Y $E.Y/$ X

N

results in a large value of its mean $E.X/$ and so we correct for this by low

(i.e. N larger than its mean xN. A similar

) then probably also N

ar
e
ne
ga
ti
ve
ly
co
rr
el
at
ed
.

ar
gu
m
en
t
ho
ld
s
w
he
n

N and N

In the simulation of the production line of section 2, a natural control variate would be the long-term average production rate for the line with zero buffers.

Conditioning

The method of conditioning is based on the following two formulas. If X and Y are two arbitrary random variables, then $E . X / D E .E . X jY //$ and Var. $X / D E$ .Var. $X jY // C$ Var.$E . X jY //$ Var.$E . X jY //$. Hence, we conclude that the random variable $E . X jY /$ has the same mean as and a smaller variance than the random variable X .

How can we use these results to reduce the variance in a simulation? Let $E . X /$ be the performance measure that we want to estimate. If Y is a random variable such that $E . X jY D y/$ is known, then the above formulas tell us that we can better simulate Y and use $E . X jY /$ than that we directly simulate X .

The method is illustrated using the example of an $M\_=M\_=1=N$ queueing model in which customers who find upon arrival N other customers in the system are lost. The performance measure that we want to simulate is $E . X /$, the expected number of lost customers at some fixed time t . A direct simulation would consist of K simulation runs until time t . Denoting by X_i the

N D P K

number of lost customers in run i , then $X . i D1 X_i /=K$ is an unbiased estimator of $E . X /$. However, we can reduce the variance of the estimator in the following way. Let Y_i be the total amount of time in the interval $.0; t /$ that there are N customers in the system in the i -th simulation

rate _, it follows that $E . X$ Y / D

run.
Since
customer
s arrive
accordin
g to a
Poisson
process
with K $Y/=K$ j i

i . Hence
an
improved
estimator
would be
_ N ,
where N D $N_{i D1}$ i t .

D 1000,
the
estimator
s

In Table 9 and Table 10 we compare, for _ D 0:5, _ D 1, P D 3 and
and confidence intervals for $E.X/$ when we do not, resp. do, use conditioning. We conclude that

33

# of runs	mean	st. dev.	95% conf. int.
10	33.10	10.06	[26.86, 39.33]
100	34.09	9.21	[32.28, 35.90]
1000	33.60	8.88	[33.05, 34.15]

Table 9: Estimation of $E.X/$ without using conditioning

# of runs	mean	st. dev.	95% conf. int.
10	33.60	7.16	[29.17, 38.05]
100	33.82	6.91	[32.46, 35.18]
1000	33.36	6.57	[32.95, 33.76]

Table 10: Estimation of $E.X/$ using conditioning

in this example the use of conditioning reduces the standard deviation of the estimator and hence also the length of the confidence interval with a factor 1.3.