Gold™

Uttarakhand Technical University

Vth Sem

One Week

Unit Wise Solution Bank for

B.TECH.

Modeling & Simulation (CS)

Unit wise Solution Bank In question Answer form | Covering Complete Syllabus
No Out of Syllabus Contents Included | Low Price to Suit Your Pocket
Including Solved University Questions

Syllabus

UNIT 1

Introduction: Systems, models, discrete event simulation and continuous simulation.
Discrete Event Simulation: Time-advance mechanisms event modeling of discrete dynamic systems, single-server single queue model, event graphs, Monte Carlo simulation.

UNIT 2

GPSS: Model structure, entities and transactions, blocks in GPSS, process oriented programming, user defined functions, SNA, logic switches, save locations, user chains, tabulation of result, programming examples.
Random Number Generation: Congruence generators, long period generators, uniformity and independence testing.

UNIT 3

Random Variate Generation: Location, scale and shape parameters, discrete and continuous probability distributions; Inverse transform method, composition and acceptance rejection methods.

UNIT 4

Queuing Models: Little's theorem, analytical results for M/M/1, M/M/1/N, M/M/c, M/G/1 and other queuing models.

UNIT 1

INTRODUCTION, DISCRETE EVENT SIMULATION

Q. 1. What is simulation? Discuss advantages of simulation? **[UTU: 2012-13]**

Ans. A simulation is the imitation of the real-world process or system over time. Whether done by hand or on a computer, simulation involves the generation of an artificial history to draw inferences concerning the operating characteristics of the real system .The behaviour of the system as it evolves over time is studied by developing a simulation model. this model usually takes the form of a set of assumptions concerning the operation of the system.

Simulation can also be used to study system in the design stage, before such systems are built. thus, simulation modelling can be used both as an analysis tool for predicting the effect of changes to existing system and as a design tool to predict the performance of new system under varying sets of circumstances.

Advantages of Simulation

1. The primary advantages of simulators are that they are able to provide users with practical feedback when designing real world systems. This allows the designer to determine the correctness and efficiency of a design before the system is actually constructed. Consequently, the user may explore the merits of alternative designs without actually physically building the systems. By investigating the effects of specific design decisions during the design phase rather than the construction phase, the overall cost of building the system diminishes significantly. As an example, consider the design and fabrication of integrated circuits. During the design phase, the designer is presented with a myriad of decisions regarding such things as the placement of components and the routing of the connecting wires. It would be very costly to actually fabricate all of the potential designs as a means of evaluating their respective performance. Through the use of a simulator, however, the user may investigate the relative superiority of each design without actually fabricating the circuits themselves. After carefully weighing the specifications of each design, the best circuit may then be fabricated.
2. Another benefit of simulators is that they permit system designers to study a problem at several different levels of abstraction. By approaching a system at a higher level of abstraction, the designer is better able to understand the behaviours and interactions of all the high level components within the system and is therefore better equipped to counteract the complexity of the overall system. As the designer better understands the operation of the higher level components through the use of the simulator, the lower level components may then be designed and subsequently simulated for verification and performance evaluation. The entire system may be built based upon this "top-down" technique. This approach is often referred to as hierarchical decomposition and is essential in any design tool and simulator which deals with the construction of complex systems. For example, with respect to circuits, it is often useful to think of a microprocessor in terms of its registers, arithmetic logic units, multiplexers and control units. A simulator which permits the construction, interconnection and subsequent

simulation of these higher level entities is much more useful than a simulator which only lets the designer build and connect simple logic gates.

3. Thirdly, simulators can be used as an effective means for teaching or demonstrating concepts to students. This is particularly true of simulators that make intelligent use of computer graphics and animation. Such simulators dynamically show the behaviour and relationship of all the simulated system's components, thereby providing the user with a meaningful understanding of the system's nature. Consider again, for example, a circuit simulator. By showing the paths taken by signals as inputs are consumed by components and outputs are produced over their respective fan out, the student can actually see what is happening within the circuit and is therefore left with a better understanding for the dynamics of the circuit. Such a simulator should also permit students to speed up, slow down, stop or even reverse a simulation as a means of aiding understanding. This is particularly true when simulating circuits which contain feedback loops or other operations which are not immediately intuitive upon an initial investigation.

Q.2. Discuss the area of application of simulation?

Ans. The applications of simulation are vast. The winter simulation conference (wsc) is an excellent way to learn more about the latest in simulation application and theory. some presentations, by area ,from a recent WSC are listed here:

1. Manufacturing applications.
2. Semiconductor manufacturing .
3. Construction Engineering and Project management.
4. Military application.
5. Logistics, Supply chain and Distribution application.
6. Transportation modes and traffic.
7. Business Process and Simulation.
8. Health care and many more.....

Q. 3. Discuss the various ways of modelling of system.

OR

What do you mean by System Modelling? What is the need of system modeling.

[UTU: 2012-13]

Ans. In business and IT development systems are modelled with different scoops and scales of complexity, such as:

- **Functional modelling:** A function model or functional model in systems engineering and software engineering is a structured representation of the functions (activities, actions, processes, operations) within the modeled system or subject area. A function model, also called an activity model or process model, is a graphical representation of an enterprise's function within a defined scope. The purposes of the function model are to describe the functions and processes, assist with discovery of information needs, help identify opportunities, and establish a basis for determining product and service costs.
- **Business process modelling:** Business Process Modelling (BPM) in systems engineering and hardware engineering is the activity of representing processes of an enterprise, so that the current process may be analyzed and improved. BPM is typically performed by business analysts and managers who are seeking to improve process efficiency and quality. The process improvements identified by BPM may or may not require Information Technology involvement, although that is a common driver for the need to model a business process, by creating a process master.

- **Enterprise modelling:** Enterprise modelling is the abstract representation, description and definition of the structure, processes, information and resources of an identifiable business, government body, or other large organization. It deals with the process of understanding an enterprise business and improving its performance through creation of enterprise models. This includes the modelling of the relevant business domain (usually relatively stable), business processes (usually more volatile), and Information technology.

Further more like systems thinking, systems modelling in can be divided into:

- **Systems analysis:** Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to requirements analysis or operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made."
- **Hard systems modelling or operational research modelling:** Operations research (also referred to as decision science, or management science) is an interdisciplinary mathematical science that focuses on the effective use of technology by organizations. In contrast, many other science & engineering disciplines focus on technology giving secondary considerations to its use.
- **Soft system modelling:** Soft systems methodology (SSM) is a systemic approach for tackling real-world problematic situations. Soft Systems Methodology is the result of the continuing action research that Peter Check land, Brian Wilson, and many others have conducted over 30 years, to provide a framework for users to deal with the kind of messy problem situations that lack a formal problem definition.

Q. 4. What is system and system environment? Explain component of the system with example.

Ans. A system is defined as a group of object that are joined together in some intersection or interdependence toward the accomplishment of some purpose .an example is a production system manufacturing automobiles. The machines, component parts, and workers operate jointly along an assembly line to produce a high-quality vehicle. A system is often affected by changes outside the system. Such changes are said to occur in the system environment. In Modelling system ,it is necessary to decide on the boundary between system and its environment. This decision may depend on the purpose of the study. A system is categorized as DISCRETE system and CONTINOUS system.

Components of system: In order to understand and analyze a system, a number of terms need to be defined:-

1. **Entity:** An entity is an object of interest in the system.
2. **Attribute:** An attribute is a property of entity.
3. **Activity:** An activity represents a time period of specified length.
4. **State:** The state of a system is defined to be that collection of variable necessary to describe the system at any time, relative to the objectives of the study
5. **Event:** An event is defined as an instantaneous occurrence that might change the state of the system.

Q. 5. What is the model of a system. Discuss the types of model?

Ans. A model is defined as a representation of a system for the purpose of studying the system. The model, by definition, is a simplification of the system. On the other hand, the model should be sufficiently detailed to permit valid conclusion to be drown about the real system. Different models of the system could be required as the purpose of investigation changes.

TYPES OF MODELS: Models can be classified as being mathematical or physical. A mathematical model uses symbolic notation and mathematical equation to represent a system. A simulation model is a particular type of mathematical model of system.

Simulation model may be further classified as:

1. STATIC: A static simulation model, sometimes called a Monte Carlo simulation, represents a system at a point in time.
2. DYNAMIC: A dynamic simulation model represent a system as they change over time. The simulation of a bank from 9:00 A.M to 4:00 P.M is an example of a dynamic simulation.
3. DETERMINISTIC: simulation model that contain no random variable are classified as deterministic. These models have a known set of I/P, which will result in a unique set of output. Deterministic arrivals would occur at a dentist office if all patient arrived at the scheduled appointment time.

Q. 6. Write short note on discrete-event simulation and continuous simulation?

OR

Discuss the concepts in discrete event Simulation. **[UTU: 2012-13]**

OR

Discuss the concepts in continuous Simulation. **[UTU: 2012-13]**

A: DISCRETE-EVENT SIMULATION: Discrete event simulation is suitable for problems in which variables change in discrete times and by discrete steps. On the other hand, continuous simulation is suitable for systems in which the variables can change continuously. Discrete-Event simulation is the modelling of system in which the state variable changes only at a discrete set of points in time . Discrete event simulation utilizes a mathematical/logical model of a physical system that portrays state changes at precise points in simulated time. Both the nature of the state change and the time at which the change occurs mandate precise description. The simulation models are analyzed by numerical method rather than by analytical method. Customers waiting for service, the management of parts inventory or military combat are typical domains of discrete event simulation.

CONTINUOUS SIMULATION: A continuous simulation is one in which the state variable(s) change continuously over time. Continuous simulation uses equational models, often of physical systems, which do not portray precise time and state relationships that result in discontinuities. The objective of studies using such models do not require the explicit representation of state and time relationships. Examples of such systems are found in ecological modelling , ballistic re-entry, or large scale economic models.

Q. 7. Differentiate between discrete-event simulation and continuous simulation?

Ans. Discrete-event

1. Discrete event simulation is suitable for problems in which variables change in discrete times and by discrete steps.
2. DES models are better at "providing a detailed analysis of systems involving linear processes and modelling discrete changes".
3. Discrete event simulation has been the major tool for arriving conclusions about about complicated queuing networks.
4. Discrete-state Markov chain, where the system state is the number of People in the system.

Continuous Simulation

1. Continuous Simulation is suitable for systems in which the variable change continuously.
2. "Problems Associated with Continuous Processes where feedback significantly affect the behaviour".
3. It is very rare to see a simulation study that uses continues simulation to analyze queuing system.
4. it is assumed that the number of people in the system and number of server busy can take continuous value.

Q. 8. Explain Time Advance Mechanism with the steps involved in it.

OR

Discuss various time events Mechanisms. **[UTU: 2012-13]**

Ans. The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the future event list (FEL). This list contains all event notices for events that have been scheduled to occur at a future time. Scheduling a future event means that, at the instant an activity begins, its duration is computed or drawn as a sample from a statistical distribution; and that the end-activity event, together with its event time, is placed on the future event list. In the real world, most future events are not scheduled but merely happen---such as random breakdowns or random arrivals. In the model, such random events are represented by the end of some activity, which in turn is represented by a statistical distribution.

At any given time t, the FEL contains all previously scheduled future events and their associated event times (called t_1, t_2, ...). The FEL is ordered by event time, meaning that the events are arranged chronologically---that is, the event times satisfy.

$$t < t_1 \leq t_2 \leq t_3 \leq \ldots\ldots t_n$$

Time t is the value of CLOCK, the current value of simulated time. The event associated with time t1 is called the imminent event; that is, it is the next event that will occur. After the system snapshot at simulation time CLOCK = t has been updated, the CLOCK is advanced to simulation time CLOCK = t_1, the imminent event notice is removed from the FEL, and the event is executed. Execution of the imminent even t means that a new system snapshot for time t1 is created, one based on the old snapshot at time t and the nature of the imminent event. At time t1, new future events may or might not be generated, but if any are, they are scheduled by creating event notices and putting them into their proper position on the FEL. After the new system snapshot for time t_1 has been updated, the clock is advanced to the time of the new imminent event and that event is executed. This process repeats until the simulation is over. The sequence of actions that a simulator (or simulation language) must perform to advance the clock and build a new system snapshot is called the event-scheduling/time-advance algorithm.

Steps involved in time advance mechanism are:

1. Remove the event notice for the imminent event from FEL.
2. Advance CLOCK to imminent event time.
3. Execute imminent event: update system state, change entity attributes, and set membership as needed.
4. Generate future events and place their event notices on FEL, ranked by, ranked by event time.
5. Update cumulative statistics and counters.

Q. 9. What are the basic elements of a queuing system?

Ans. In order to model queuing systems, we first need to be a bit more precise about what constitutes a queuing system. The three basic elements common to all queuing systems are:

Arrival Process: Any queuing system must work on something--customers, parts, patients, orders, etc. We generically term these entities. Before entities can be processed or subjected to waiting, they must first enter the system. Depending on the environment, entities can arrive smoothly or in an unpredictable fashion. They can arrive one at a time or in clumps (e.g., bus loads or batches). They can arrive independently or according to some kind of correlation.

Service Process: Once entities have entered the system they must be served. The physical meaning of "service" depends on the system. Customers may go through the checkout process. Parts may go through machining. Patients may go through medical treatment. Orders may be filled. And so on. From a modeling standpoint, the operational characteristics of service matter more than the physical characteristics. Specifically, we care about whether service times are long or short, and whether they are regular or highly variable. We care about whether entities are processed in first-come-first-serve (FCFS) order or according to some kind of priority rule. We care about whether entities are serviced by a single server or by multiple servers working in parallel.

Queue: The third required component of a queuing system is a queue, in which entities wait for service. The simplest case is an unlimited queue which can accommodate any number of customers. But many systems (e.g., phone exchanges, web servers, call centers), have limits on the number of entities that can be in queue at any given time. Arrivals that come when the queue is full are rejected (e.g., customers get a busy signal when trying to dial into a call center). Even if the system doesn't have a strict limit on the queue size, customers may balk at joining the queue when it is too long (e.g., cars pass up a drive through restaurant if there are too many cars already waiting). Entities may also exit the system due to impatience (e.g., customers kept waiting too long at a bank decide to leave without service) or perish ability (e.g., samples waiting for testing at a lab spoil after some time period).

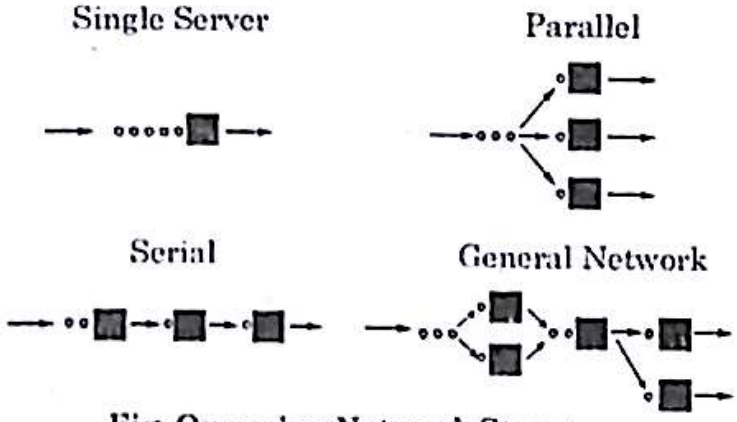


Fig. Queueing Network Structure

Q. 10. Explain briefly the single server single queue model with example.

Ans. Single Server - Single Queue: The models that involve one queue - one service station facility are called single server models where customer waits till the service point is ready to take him for servicing. Single-server single queues are, perhaps, the most commonly encountered queuing situation in real life. One encounters a queue with a single server in many situations, including business (e.g. sales clerk), industry (e.g. a production line), transport (e.g. a queues that the customer can select from.) Consequently, being able to model and analyse a single server queue's behaviour is a particularly useful thing to do.

The single server system, is the simplest case and represents many real world settings such as the checkout lane in a drugstore with a single cashier or the CPU of a computer that processes jobs from multiple applications.

Students arriving at a library counter are an example of a single server facility.

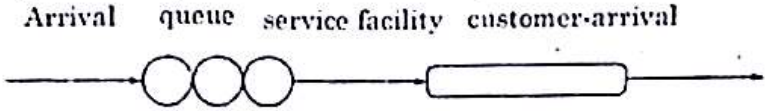


Fig. Single Server - Single Queue Model

Q.11. What are Event Graphs? Explain their importance in Discrete- event simulation.

Ans. Event Graphs are a way of graphically representing discrete-event simulation models. Also known as "Simulation Graphs," they have a minimalist design, with a single type of node and two types of edges with up to three options. Despite this simplicity, Event Graphs are extremely powerful. The Event Graph is the only graphical paradigm that directly models the event list logic. There are no limitations to the ability of Event Graphs to create a simulation model for any circumstance. Their simplicity, together with their extensibility, make them an ideal tool for rapid construction and prototyping of simulation models.

An event graph partitions the model into events, represented by vertices, and uses directed edges to represent relationships between events. We define an event to be the largest unit of a program that will execute atomically during simulation, which corresponds to the code that would be executed during one step of an event-driven simulation. The semantics of the original language determines precisely how the boundaries between events are determined from the original source.

Event Graphs are a way of representing the Future Event List logic for a discrete-event model. An Event Graph consists of nodes and edges. Each node corresponds to an event, or state transition, and each edge corresponds to the scheduling of other events. Each edge can optionally have an associated Boolean condition and/or a time delay. Below figure shows the fundamental construct for Event Graphs and is interpreted as follows: the occurrence of Event A causes Event B to be scheduled after a time delay of *t*, providing condition (i) is true (after the state transitions for Event A have been made). By convention, the time delay t is indicated toward the tail of the scheduling edge and the edge condition is shown just above the wavy line through the middle of the edge. If there is no time delay, then t is omitted. Similarly, if Event B is always scheduled following the occurrence of Event B, then the edge condition is omitted, and the edge is called an unconditional edge.

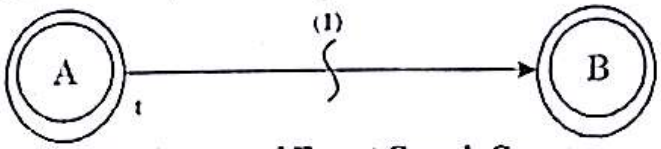


Fig. Fundamental Event Graph Construct

Thus, the basic Event Graph paradigm contains only two elements (event node and scheduling edge) with two options on the edges (time delay and edge condition). The simplicity of the Event Graph paradigm is evident from the fact that we can represent any discrete event model using only these constructs.

An important feature of event graphs in DES is parameterization of the event vertices, allowing similar model sub-graphs to be combined together as a generic sub-graph distinguished by parameter values. A framework based on an Integrated Entity/Event (IE2) approach has been further enhanced to explicitly represent entities at the event-driven level. The integrated simulation framework works towards attenuation of the abstraction involved in parameter passing. The solution lies in explicitly passing the entities through the event-driven model. Event parameters are replaced by entity attributes. The usage of entities in the event-driven layer serves two purpose, (a) reduces the abstraction by manipulating entity objects instead of working with parameters as in a programming

language, and (b) gives the intuitive feel of process-driven models to modellers at the event level, which enhances the appeal of the event-driven models.

Q. 12. Explain briefly the Event modelling of discrete system.

Ans. The great majority of processes we observe in the world consist of continuous changes. However, when we try to analyze these processes it often makes sense to divide a continuous process into discrete parts to simplify the analysis. Discrete Event Modelling techniques approximate continuous real-world processes with non-continuous events that you define.

Here are some examples of events:

- A customer arrives at a shop,
- A truck finishes unloading,
- A conveyor stops,
- A new product is launched,
- Inventory levels reaches a certain threshold, etc.

In discrete event modelling the movement of a train from point A to point B would be modeled with two events, namely a departure event and an arrival event. The actual movement of the train would be modeled as a time delay (interval) between the departure and arrival events. This doesn't mean however that you can't model the train as moving. In fact, with Any Logic you can produce visually continuous animations for logically discrete events Discrete event modelling techniques should be used only when the system under analysis can naturally be described as a sequence of operations. It is not always clear for any given modelling project which of the three modelling paradigms is best. For example, if it is easier to describe the behaviour of each individual entity than trying to put together a global workflow, agent based modelling may be the way to go. Similarly, if you are interested in aggregates and not in individual unit interaction, system dynamics may be applied. Any Logic supports all three modelling approaches, so you can experiment with the abstraction levels and modelling approach without needing multiple tools.

Q. 13. What is Monte Carlo simulation? Explain.

OR

Monte Carlo Simulation is a special case of stochastic simulation? Comment. [UTU: 2012-13]

Ans. What Does Monte Carlo Simulation Mean?

A problem solving technique used to approximate the probability of certain outcomes by running multiple trial runs, called simulations, using random variables.

Monte Carlo simulation is named after the city in Monaco, where the primary attractions are casinos that have games of chance. Gambling games, like roulette, dice, and slot machines, exhibit random behaviour.

Monte Carlo simulation is a computerized mathematical technique that allows people to account for risk in quantitative analysis and decision making. The technique is used by professionals in such widely disparate fields as finance, project management, energy, manufacturing, engineering, research and development, insurance, oil & gas, transportation, and the environment.

Monte Carlo simulation furnishes the decision-maker with a range of possible outcomes and the probabilities they will occur for any choice of action.. It shows the extreme possibilities-the outcomes of going for broke and for the most conservative decision-along with all possible consequences for middle-of-the-road decisions.

Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values-a probability distribution-for any factor that has inherent uncertainty. It then calculates results over and over, each time using a different set of random values from the probability functions. Depending upon the number of uncertainties and the ranges specified for them, a Monte Carlo simulation could involve thousands or tens of thousands of recalculations before it is complete. Monte Carlo simulation produces distributions of possible outcome values.

By using probability distributions, variables can have different probabilities of different outcomes occurring. Probability distributions are a much more realistic way of describing uncertainty in variables of a risk analysis. Common probability distributions include:

Normal: Or "bell curve." The user simply defines the mean or expected value and a standard deviation to describe the variation about the mean. Values in the middle near the mean are most likely to occur. It is symmetric and describes many natural phenomena such as people's heights. Examples of variables described by normal distributions include inflation rates and energy prices.

Lognormal: Values are positively skewed, not symmetric like a normal distribution. It is used to represent values that don't go below zero but have unlimited positive potential. Examples of variables described by lognormal distributions include real estate property values, stock prices, and oil reserves.

Uniform: All values have an equal chance of occurring, and the user simply defines the minimum and maximum. Examples of variables that could be uniformly distributed include manufacturing costs or future sales revenues for a new product.

Triangular: The user defines the minimum, most likely, and maximum values. Values around the most likely are more likely to occur. Variables that could be described by a triangular distribution include past sales history per unit of time and inventory levels.

PERT: The user defines the minimum, most likely, and maximum values, just like the triangular distribution. Values around the most likely are more likely to occur. However values between the most likely and extremes are more likely to occur than the triangular; that is, the extremes are not as emphasized. An example of the use of a PERT distribution is to describe the duration of a task in a project management model.

Discrete: The user defines specific values that may occur and the likelihood of each. An example might be the results of a lawsuit: 20% chance of positive verdict, 30% change of negative verdict, 40% chance of settlement, and 10% chance of mistrial.

During a Monte Carlo simulation, values are sampled at random from the input probability distributions. Each set of samples is called an iteration, and the resulting outcome from that sample is recorded. Monte Carlo simulation does this hundreds or thousands of times, and the result is a probability distribution of possible outcomes. In this way, Monte Carlo simulation provides a much more comprehensive view of what may happen. It tells you not only what could happen, but how likely it is to happen.

The Monte Carlo method is just one of many methods for analyzing uncertainty propagation, where the goal is to determine how random variation, lack of knowledge, or error affects the sensitivity, performance, or reliability of the system that is being modeled. Monte Carlo simulation is categorized as a sampling method because the inputs are randomly generated from probability distributions to simulate the process of sampling from an actual population. So, we try to choose a distribution for the inputs that most closely matches data we already have, or best represents our current state of knowledge. The data generated from the simulation can be represented as probability distributions

(or histograms) or converted to error bars, reliability predictions, tolerance zones, and confidence intervals. (See Figure).

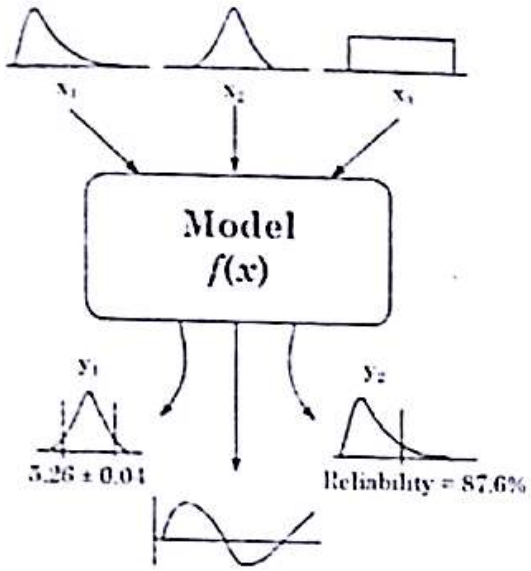


Fig. Schematic showing the principal of stochastic uncertainty propagation. (The basic principle behind Monte Carlo simulation.)

The steps in Monte Carlo simulation corresponding to the uncertainty propagation shown in Figure are fairly simple, and can be easily implemented in Excel for simple models. All we need to do is follow the five simple steps listed below:

Step 1: Create a parametric model, $y = f(x_1, x_2, ..., x_q)$.

Step 2: Generate a set of random inputs, $xi_1, xi_2, ..., x_{iq}$.

Step 3: Evaluate the model and store the results as y_i.

Step 4: Repeat steps 2 and 3 for $i = 1$ to n.

Step 5: Analyze the results using histograms, summary statistics, confidence intervals, etc.

Q.14. Differentiate between monte Carlo computation and stochastic simulation.

[UTU: 2011-12]

Ans. Monte carlo simulation is used to give solutions of deterministic problems whereas stochastic simulation is used for stochastic problems. Basically Monte carlo simulation was named after world war-2 by j. Von Newmann to solve real world problems.

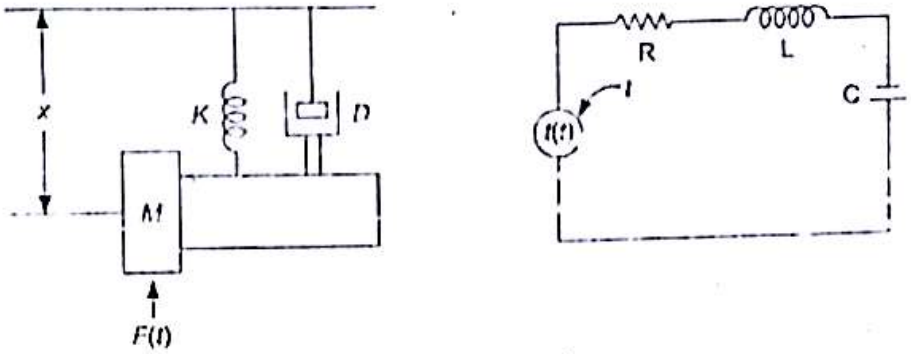
Monte carlo computation solves deterministic problems. Any simulation model that does not contain any random or probabilistic element is called a deterministic simulation mode. The characteristic of this type of simulation model is that the output is determined when the set of input elements and properties in the model have been specified. For example, a deterministic simulation model can represent a complicate system of differential equations.

Many simulation models however, have at least one element that is random, which gives rise to the stochastic simulation mode. In most simulation models randomness is important to mimic the real scenario, for example user connections to the internet arise 'randomly' when a person pressing a key. However, for any stochastic simulation model that has random output, the output (numerical results) can only be treated as an estimate of the true output parameters of the model.

Q.15.What is difference between static physical models and dynamic physical models? Explain with suitable example. **[UTU: 2011-12]**

Ans. A static physical model is used as a means of solving equations with particular boundary condition. These are many examples in the field of mathematical physics where the same equations apply to different physical phenomena. For example the flow of heat and the distribution of electronic charge through space can be related by common equation.

Dynamic physical models rely upon an analogy b/w the system being studied and some other system of a different nature, the analogy usually depending upon an underlying similarity in the focus governing the behaiour of systems.



(a) Mechanical (b) Electrical system.

In (a) the motion of the system is described by the following differential equation.

$$M\ddot{x} + D\dot{x} + Kx = kF(t)$$

In (b) behaviour of the circuit is governed by the following equation;

$$L\ddot{q} + R\dot{q} + \frac{q}{C} = \frac{E(t)}{C}$$

Inspection of these two equations shows that they have exactly the same form and the following equations occur b/w the quantities in the two systems.

Displacement x	Charge q
Velocity	current $I = \ddot{q}$
Force F	Voltage E
Mass M	Inductance L
Damping factor D	Resistance R
Spring Stiffness K	I/capacitance I/C

Q.16. Differentiate between next - even time advance approach and fixed event time advance approach. **[UTU: 2011-12]**

Ans. In simulating any dynamic system-continuous or discrete—there must be a mechanism for the flow of time.

In fixed time step model a "timer" or "clock" is simulated by the computer. This clock is updated by fixed time interval Z, and the system is examined to see if any event has taken place during this time interval (minutes) hours and days whatever) all events that take place during this period are treated as if they occurred simultaneously at the tail end of this interval.

In a next-event simulation model the computer advances time to the occurrence of the next event. It shifts from event to event. The system state does not change in between.

Q.17. "Simulation is the process of carrying out sampling experiments on the model of the system than the system itself." Discuss. **[UTU: 2011-12]**

Ans. Simulation is basically an experimental technique. It is a fast and relatively inexpensive method of doing an experiment on the model of system than the system itself. For example—the inventory control problem. Instead of trying out in the actual store the five replenishment policies, each for a period of six month, and then selecting the best one, we conducted the experiment on computer and obtained the result within few minutes.

Q.18. What are major limitations of simulation? **[UTU: 2011-12, 2012-13]**

Ans. These are the major limitations in simulation:

Expensive to build a simulation model.

Expensive to conduct simulation.

Sometimes it is difficult to interpret the simulation results.

Simulation concerns the manipulation of a number of variables of a model representing a real system. However, manipulation of a single variable often means that the reality of the system as a whole can be lost. Certain systems of components of a realistic situation are not transparent. Some factors have a lot of influence on the whole, but they have indistinct relations in the whole and can therefore not be represented in a model. These factors, however, cannot be forgotten in the learning process.

A computer simulation program cannot develop the students's emotional and intuitive awareness that the use of simulation is specifically directed at establishing relations between variables in a model. So this intuition has to be developed in a different way.

Computer simulation cannot reach to unexpected 'sub-goals' which the student may develop during a learning-process. They sub-goals would be brought up during a teacher-student interaction but they remain unsaid during the individual student use of a simulation.

During the experience of interaction with a computer simulation program, the student is frequently asked to solve problems in which creativity is often the decisive factor to success. The fact that this creativity is more present in some pupils than in other is not taken into account by the simulation. Mutual collaboration and discussion among students while using the software could be a solution for this.

❑❑❑

UNIT

2

GPSS, RANDOM NUMBER GENERATION

2.1. Explain briefly the General Purpose Simulation System (GPSS).

Ans. General Purpose Simulation System (GPSS) is a discrete time simulation language, where a simulation clock advances in discrete steps. A system is modeled 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. It was popular in the late 1960s and early 1970s but is little used today. General Purpose Simulation System is a language designed for discrete event modeling. The language as it exists today is the result of more than twenty years of evolution. While GPSS has its roots in the early days of mainframe computing, its basic ideas have proven suitable for application to today problems using modern computing environments. The popularity of GPSS is due, in part, to its power of expression. A short, easily understood GPSS model would require many pages of FORTRAN coding to accomplish a similar goal. The GPSS user is free to concentrate on the important issues in the model being developed since the language itself collects statistics, produces tabulated results and performs a host of mundane tasks one would prefer not to deal with.

GPSS provides a set of abstract components of various types and a set of operators called blocks which perform certain actions on the individual components. The transaction is the component which moves through a sequence of blocks that has been designed to model the system being studied. The state of the components of the model determines the details of how a block of a given type will operate. For example, a block which permits a transaction to take control of a piece of equipment will not allow the transaction to proceed if the equipment is already at maximum capacity. GPSS provides an underlying control mechanism which is transparent to the user but which ensures that competition between transactions (for use of a facility, for instance) is consistently arbitrated and that transactions are moved through the blocks of the model in an orderly and efficient way. This control mechanism also administers the GPSS clock which provides the time-base for models. The GPSS clock is also an abstraction. It measures time in clock units. A clock unit is interpreted appropriately by the person designing the model. In the factory model a clock unit could correspond to a second or a minute, while in the communication network the finer resolution of a one- millisecond clock unit would be more suitable.

2. 2. Describe briefly the modeled structure of GPSS.

OR

Discuss any five blocks in GPSS. **[UTU: 2012-13]**

Ans. The functionality and modelled structure of GPSS in simulation can be described by the example of bank given below .

The following brief example models a bank with a individual queue. The transactions are customers and facilities are used to represent the bank tellers.

```
1.                SIMULATE
2.   LINES        EQU          1(5),Q,F
3.   ARRIVE       FUNCTION     RN5,BE
4.                GENERATE     20,FNSARRIVE
```

```
5.    SELECT MIN    1,LINES,LINES+4,,Q
6.    QUEUE         P1
7.    SEIZE         P1
8.    DEPART        P1
9.    ADVANCE       90,20
10.   RELEASE       P1
11.   TERMINATE     1
12.   START         100
13.   END
```

The following is a line-by-line description of the model:

1. SIMULATE is a control statement.
2. EQU defines LINES as the name of the first of a group of 5 corresponding queues and facilities - the lineups and tellers.
3. Defines ARRIVE as the name of a function transforming a random number between 0 and 1, into a value from a built in exponential ("BE") distribution - a commonly-used in simulating arrival processes. The random number is taken from the fifth built in stream of such numbers ("RN5").
4. Customers arrive an average of 20 seconds apart with a random variation determined by sampling the function ARRIVE defined in the previous line.
5. Set parameter 1 of each entering transaction to the number of the queue with the minimum length.
6. Join the queue whose number is now in parameter 1.
7. When the corresponding facility is available take possession of it.
8. Now leave the lineup, recording the delay time in the queue statistics.
9. Delay for an average of 90 seconds with a variation of 20 seconds either way, representing time for the banking to be done. The delay is chosen randomly with uniform probability in the 70 to 110 second range. The facility remains in use for this period. The random number used to determine the actual delay time is taken from the first random number stream, thus the arrival times and delay times are statistically independent.
10. Free the facility for use by the next transaction.
11. Destroy the transaction. Count 1 complete customer cycle.
12. Run the model until 100 customers have reached the TERMINATE block.
13. END marks the last line of the model.

Q. 3. What is the use of GPSS in simulation? Discuss some of its application.

Ans. The main use or advantage of GPSS and other similar environment(process oriented simulation), is that the simulation software take care of managing the interaction among the processes. There is no need for the program developer to be concerned with the details. But structure diagrams are useful for keeping the program developer mindful of this underlying processing when creating block segment. the next section outlines how the GPSS blocks are used to implement processes. GPSS is highly structured ,special purpose simulation programming language based on the process interaction approach and oriented toward queueing system. A block diagram provides a convenient way to describe the system being simulated. GPSS can be used to model any situation where transaction are flowing through a system . the block diagram is converted to block statement , control statement are added , and the result is a GPSS model. GPSS/H continues to be one of the most general, flexible, and powerful simulation environments currently available. GPSS/H is presently applied worldwide to model manufacturing, transportation, distribution

telecommunications, hospitals, computers, logistics, mining, and many other types of queuing systems. Multiple transactions can execute GPSS/H model statements at the same instant in (simulated) time without any special action required of the modeller . The execution of a process-interaction simulation model is thus similar to a multi-threaded computer program. This differs greatly from the single-threaded, sequential execution of most general-purpose programming languages.

Q. 4. Discuss briefly the role of Entities in GPSS.

Ans. GPSS is built around several elementary abstractions called entities. In order for you to be able to create complex simulation, you must acquire an understanding of these entities and of the rules by which they may be manipulated. GPSS entities are abstract objects that exist in a simulation. The collection of all the entities is called the simulation. The most prominent entity types are Transactions and Blocks, because simulations, to a large extent, consist of many Transactions moving from one Block into the next. Transactions are the only entity types that can be deleted from the simulation. In all, there are a dozen or so entity types, and a simulation may contain many instances of any entity type. To be effective in creating simulations you must understand the properties of each of the GPSS entities and how to use GPSS Blocks in order to cause interactions among the entities.

GPSS entities are numbered. When you use a name to refer to an entity, the integer value associated with the name is used to find the entity. However you do not normally assign these integer values to names, although the EQU statement enables you to do so. GPSS World normally assigns a unique value greater than or equal to 10,000. Most GPSS entities are created automatically when needed. For example, a reference to a Facility Entity using the name Barber will cause a Facility to be created if none existed before some entities must be specifically declared before they can be used. Generally these have an attribute, such as size, which must be made known to the Simulation Object. The name in the label field, called an Entity Label, is then used to refer to the entity.

The following entities must be declared before they can be used:

- Storage entities must be declared in STORAGE Statements.
- Arithmetic Variables must be declared in VARIABLE Statements.
- "Floating point" Variables must be declared in FVARIABLE Statements.
- Boolean Variables must be declared in BVARIABLE Statements.
- Matrices must be declared in MATRIX Statements, or Temporary Matrix PLUS Declarations.
- Tables must be declared in TABLE Statements.
- Qtables must be declared in QTABLE Statements.
- Functions must be declared in FUNCTION and Function Follower Statements.
- Transaction Parameters must be declared in ASSIGN, MARK, READ, SELECT, SPLIT, COUNT or TRANSFER SUB Blocks before they are referenced.

Q. 5. Define the term Transaction and the transaction attribute used in GPSS.

Ans. Transactions move from Block to Block in a manner which represents the real-world system you are modeling. Once a Transaction begins to move in the simulation, it continues to enter Blocks as long as it can. During a simulation, the Transaction which is attempting from Block to Block is called the Active Transaction. If a Transaction fails to find favorable conditions when it attempts to enter a Block, it may come to rest. Then, another Transaction is chosen to begin to move through the simulation until it, in turn, comes to rest.

The behavior of a Transaction is determined somewhat by several state variables called Transaction Attributes. The important attributes of Transactions are:

- **Parameters:** Transaction Parameters are a set of values associated with a Transaction. Each Transaction may have any number of Parameters. Each Parameter has a Parameter Number, by which it is referenced, and a value. The Parameter Number is a positive integer. The value of any Parameter of the active Transaction may be returned by the SNA PParameter where Parameter is the name or number of the Parameter. If a Block operand specifies Parameter name or number as the desired value, PSParameter or PParameter should not be used, use only the Parameter Name or Number by itself.
- **Priority:** The priority of a Transaction determines the preference it receives when it and other Transactions are waiting for the same resource. Transactions with higher priority values receive preference. The most important priority queues in the simulation are the Current Events Chain, Facility Delay Chains, and Storage Delay Chains. The Future Events Chain is not a priority chain. The effect of priority is that a Transaction will be chosen ahead of lower priority Transactions when a new Active Transaction, or a new Facility or Storage owner must be chosen. Transactions within a priority are usually scheduled first come, first served.
- **Mark Time:** The absolute clock time that the Transaction first entered the simulation or entered a MARK Block with no A operand.
- **Assembly Set:** A positive integer kept internally in each Transaction. Assembly Sets are used to synchronize Transaction in ASSEMBLE, GATHER, and MATCH Blocks. When a Transaction is created by a GENERATE Block, its Assembly Set is set equal to its Transaction Number. When a Transaction is created by a SPLIT Block, its Assembly Set is set equal to that of the parent Transaction. A Transaction can modify its Assembly Set by entering an ADOPT Block.
- **Delay Indicator:** A flag kept in each Transaction that is set by Block entry refusal, and is reset by entry into TRANSFER SIM Block. It is used by TRANSFER SIM Blocks to redirect Transactions.
- **Trace indicator:** A flag kept in each Transaction that causes a trace message to be generated each time the Transaction enters a Block. The Trace Indicator is set by a TRACE Block and reset by an UNTRACE Block.
- **Active:** The Transaction is the highest priority Transaction on the Current Events Chain.
- **Sespended:** The Transaction is waiting on the Future Events Chain or the Current Events Chain to become the active Transaction.
- **Passive:** The Transaction has come to rest in the simulation on a User Chain, Delay Chain, or Pending Chain.
- **Terminated:** The Transaction has been destroyed and no longer exists in the simulation.
- **Chains:** The state of a Transaction is determined to some degree by the chains on which it resides.

Q. 6. Explain the concept and role of Blocks in GPSS.

Ans. We first note that block diagrams have been important for the learning of GPSS and have been used in most GPSS text books. The full value of block diagrams is obtained first when they can be automatically generated from program code and that block diagrams in turn can generate code. The computer generation of block diagrams requires, however, very exact rules for the block diagrams. This is lacking in the literature. We here present the first steps towards a definition of block diagram rules that can be the basis for GPSS block diagram generation, in particular for micro-GPSS.

GPSS has all since its first version in 1961 been using some kind of block diagram symbols. Although the symbols used and the way the block symbols have been connected have changed from book to book, several symbols have remained the same over the years. the GPSS block diagram is

however, one of the major competitive advantages of GPSS. It has many strong competitors when it comes to text based simulation languages, but the GPSS block diagrams have unique features.

The full advantage of the GPSS block diagram system can, however, be obtained first when the block diagrams are generated automatically from the program text, and, even more importantly, the program text is generated from a block diagram that the user builds using a mouse. The earliest attempts of automatically generating a block diagram from the text are probably that of GPSS/PC from the early 80's.

Q. 7. What are the advantages and disadvantages of User Define Function.

Ans. Advantages of User Define Function are:

- **They allow modular programming:** You can create the function once, store it in the database, and call it any number of times in your program. User-defined functions can be modified independently of the program source code.
- **They allow faster execution:** Similar to stored procedures, Transact-SQL user-defined functions reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated executions. This means the user-defined function does not need to be reparsed and reoptimized with each use resulting in much faster execution times. CLR functions offer significant performance advantage over Transact-SQL functions for computational tasks, string manipulation, and business logic. Transact-SQL functions are better suited for data-access intensive logic.
- **They can reduce network traffic:** An operation that filters data based on some complex constraint that cannot be expressed in a single scalar expression can be expressed as a function. The function can then invoked in the WHERE clause to reduce the number or rows sent to the client.

Disadvantages of User Defined Functions:

- User Defined Functions cannot be used to modify base table information. The DML statements INSERT, UPDATE, andDELETE cannot be used on base tables.
- Another disadvantage is that SQL functions that return non-deterministic values are not allowed to be called from inside User Defined Functions. GETDATE is an example of a non-deterministic function. Every time the function is called, a different value is returned. Therefore, GETDATE cannot be called from inside a UDF you create.

Q. 8. Describe components and types of User Defined Function.

Ans. All user-defined functions have the same two-part structure: a header and a body. The function takes zero or more input parameters and returns either a scalar value or a table.

The header defines:

- Function name with optional schema/owner name
- Input parameter name and data type
- Options applicable to the input parameter
- Return parameter data type and optional name
- Options applicable to the return parameter

The body defines the action, or logic, the function is to perform. It contains either:

- One or more Transact-SQL statements that perform the function logic
- A reference to a .NET assembly

Types of User Defined Functions: There are three different types of User Defined Functions. Each type refers to the data being returned by the function. Scalar functions return a single value. In Line Table functions return a single table variable that was created by a select statement. The final UDF is a Multi-statement Table Function. This function returns a table variable whose

structure was created by hand, similar to a Create Table statement. It is useful when complex data manipulation inside the function is required.

1. Scalar UDFs: Our first User Defined Function will accept a date time, and return only the date portion. Scalar functions return a value. From inside Query Analyzer, enter:

```
AS
CREATE FUNCTION dbo.DateOnly(@InDateTime datetime)
RETURNS varchar(10)
BEGIN
   DECLARE @MyOutput varchar(10)
   SET @MyOutput = CONVERT(varchar(10),@InDateTime,101)
   RETURN @MyOutput
END
```

To call our function, execute: SELECT dbo.DateOnly(GETDATE()

Notice the User Defined Function must be prefaced with the owner name, DBO in this case. In addition, GETDATE can be used as the input parameter, but could not be used inside the function itself. Other built in SQL functions that cannot be used inside a User Defined Function include RAND, NEWID, @@CONNCECTIONS, @@TIMETICKS, and @@PACK_SENT. Any built in function that is non-deterministic.

The statement begins by supplying a function name and input parameter list. In this case, a date time value will be passed in. The next line defines the type of data the UDF will return. Between the BEGIN and END block is the statement code. Declaring the output variable was for clarity only. This function should be shortened to:

```
CREATE FUNCTION testDateOnly(@InDateTime datetime)
RETURNS varchar(10)
AS
BEGIN
    RETURN CONVERT(varchar(10),@InDateTime,101)
END
```

2. Inline Table UDFs: These User Defined Functions return a table variable that was created by a single select statement. Almost like a simply constructed non-updatable view, but having the benefit of accepting input parameters.

This next function looks all the employees in the pubs database that start with a letter that is passed in as a parameter. In Query Analyzer, enter and run:

```
USE pubs
GO
CREATE FUNCTION dbo.LookByFName(@FirstLetter char(1))
RETURNS TABLE
AS
RETURN SELECT *
FROM employee
WHERE LEFT(fname, 1) = @FirstLetter
```

To use the new function, enter:

```
SELECT * FROM dbo.LookByFName('A')
```

All the rows having a first name starting with A were returned. The return is a Table Variable, not to be confused with a temporary table. They are a special data type whose scope is limited to the process that declared it. Table variables are stated to have performance benefits over temporary tables. None of my personal testing has found this result though.

3. Multi Statement UDFs: Multi Statement User Defined Functions are very similar to Stored Procedures. They both allow complex logic to take place inside the function. There are a number of restrictions unique to functions though. The Multi Statement UDF will always return a table variable-and only one table variable. There is no way to return multiple result sets. In addition, a User Defined Function cannot call a Stored Procedure from inside itself. Remember also, that UDFs cannot use non-deterministic built in functions. So GETDATE and RAND cannot be used. Error handling is restricted. RAISERROR and @@ERROR are invalid from inside User Defined Functions. Like other programming languages, the purpose of a User Defined Function is to create a stand-alone code module to be reused over and over by the global application.

For a Multi Statement test, we will create a modified version of the LookByFName function. This new function will accept the same input parameter. But rather than return a table from a simple select, a specific table will be created, and data in it will be manipulated prior to the return:

```
CREATE FUNCTION dbo.multi_test(@FirstLetter char(1))
RETURNS @Result TABLE
      (
      fname varchar(20),
      hire_date datetime,
      on_probation char(1)
      )
AS
BEGIN
      INSERT INTO @Result
            (fname, hire_date)
            SELECT fname, hire_date
            FROM employee
            WHERE LEFT(fname, 1) = @FirstLetter

      UPDATE @Result
      SET on_probation = 'N'

      UPDATE @Result
      SET on_probation = 'Y'
      WHERE hire_date < '01/01/1991?
     RETURN
END
```

Q.9. Give the brief description on Standard Numerical Attributes (SNAs).

Ans. Every time a transaction encounters a certain block such as a QUEUE, ENTER or SEIZE block, certain statistics are gathered and kept for printing out at the end of the program. These are known as Standard Numerical Attributes (SNA's) and can be used by the programmer in other blocks when the program is being run. When the QUEUE, SEIZE, and ENTER blocks were introduced, the various SNA's associated with them were also given but not used much to this point. There are, however, many uses for them and these will become apparent as more GPSS blocks are presented. For example, the length of a queue may be used to see if a transaction will enter the queue or not; a facility is either in use or idle. This is denoted by a 1 or a 0. If a facility is being used, a transaction may be sent to a different block. As another facility is used by more and more transactions, the speed at which it operates may decrease. We shall learn how to use these

SNA's to greatly increase our programming skills.

To illustrate what some of the SNA's are, consider a portion of a program having a queue, a facility and a storage as follows. The program has to do with cars arriving for minor service at a repair shop that has three service bays. The arrival rate is a car every 100 ± 23 seconds. The cars are all first inspected by a single inspector who takes 50 ± 6 seconds to inspect the cars. The minor repairs are done in only 25 ± 7 seconds.

```
LINE            FORM        QUEUE FOR INSPECTION
SEIZE           BILL        BILL IS THE INSPECTOR
DEPART          LINE        LEAVE THE QUEUE
ADVANCE         50,6        BILL DOES THE INSPECTION
RELEASE         BILL        FREE BILL
SECND ENTER     REPAIR      THREE SERVICE BAYS
AVAILABLE
THIRD ADVANCE   25,7        NEXT SERVICE  SIMULATE
STORAGE         S(REPAIR),3 THREE SERVICE BAYS
GENERATE        100,23 CARS COME FOR REPAIRS
                            FIRST QUEUE
LEAVE           REPAIR      LEAVE STATION
```

Any SNA can be used in a program as an operand. The use of SNA's will greatly expand one's ability to build meaningful simulation models. As additional blocks are introduced, it will become even more apparent how useful they are in writing simulation models. A rich set of Standard Numerical Attributes (SNA's) provide a convenient notational way to access components of a model. For instance, Q$BOXOFFICE represents the current contents of the queue entity named BOXOFFICE. This SNA may be used wherever a numeric value is allowed. A simple balking situation could be represented by the block.

TEST L Q$BOXOFFICE,FN$BALK,GOAWAY: Which allows transactions to pass through if the current length of the BOXOFFICE lineup is less than the value returned by the function BALK, and otherwise re-directs them to the blockGOAWAY. BALK may be written to return a random variate from any desired probability distribution; or it may return a value varying with the time of day in the model or a value dependent on the number of packages (represented by a transaction parameter) the theatre patron is carrying.

Q. 10. Explain the use and requirement of logic switches in GPSS with example.

Ans. Consider the following blocks:

```
    TEST E    XH(LOCK),0
    ..... .....
    ..... .....
    GENERATE    ,,1
BACK ADVANCE  RVEXPO(1,125)
SAVEVALUE     LOCK,1,XH
ADVANCE       RVEXPO(1,12.5)
SAVEVALUE     LOCK,0,XH
TRANSFER      BACK
```

When the program begins, the value of the half word SAVEVALUE LOCK is 0 (as are all save values unless the INITIAL statement is used to specify initial values) so the transactions that enter the TEST block will pass through to the next sequential block. The transaction created in the GENERATE block is put on the FEC chain for a time given by sampling from the exponential distribution with a mean of 125. After this transaction is returned to the CEC, the value of the

SAVEVALUE LOCK is set equal to 1 (any other value could have been selected for the SAVEVALUE for this example). The transaction is then put on the FEC for a time given by sampling from the exponential distribution with a mean of 12.5. Now, any transactions which enter the TEST block will be delayed until the value of the SAVEVALUE LOCK becomes equal to 0. This delay might represent a traffic light turning red, a break in a factory for lunch, a breakdown of an assembly line, etc.

Rather that use SAVEVALUE's and TEST blocks in this manner, there is a better way to handle such conditions in GPSS. This is by using switches that are either "on" or "off". The "on" condition is known as "set" and the "off" as "reset". These switches are turned on and off by the LOGIC block. Its form is as follows:

LOGIC (R) (name or number of switch)
where R is a logic relationship and is either:
S for set
R for reset
I for invert

When a transaction enters a logic block, the effect is as follows:

(a) If the logic relationship is R, the switch is put in a reset position.
(b) If the logic relationship is S, the switch is put in a set position.
(c) If the logic relationship is I, the switch is inverted, i. e., if it was set, it becomes reset and vice versa.

Examples of these might be:
LOGIC S HALT
LOGIC R 1
LOGIC I WAIT
Notice that logic switches can be named either symbolically or numerically.

Q. 11. Explain in brief the term "User Chains".

Ans. Each User chain Entity contains a Transaction chain called a User Chain. For a more detailed explanation of the entity type, User chain,. Here we discuss the Transaction chain, called the User Chain, which is contained in each User chain entity. User chains are linked lists of Transactions which have been removed from the Current Events Chain by a LINK Block. Traditionally, there have been two uses for User Chains.

- First, it is possible to implement extremely complex Transaction scheduling disciplines with User Chains. This can be done by assigning a numerical order value to a Transaction Parameter before LINKing the Transaction on a User Chain.
- Second, older implementations of GPSS suggest that User Chains be used to avoid scheduling inefficiencies in the GPSS processor. This is less true in GPSS World because blocked Transactions do not remain on the CEC in GPSS World. However, it is still more efficient to avoid testing conditions which cannot possibly result in a successful test. In this case, you can place the blocked Transaction(s) on a User Chain until there is a possibility of success.

Q. 12. Write short note on Tabulation of result.

Ans. Micro simulation models typically produce two types of output:

1. Animation displays and
2. Numerical output in text files.

The animation display shows the movement of individual vehicles through the network over the simulation period. Text files report accumulated statistics on the performance of the network.

It is crucial that the analyst reviews both numerical and animation output (not just one or the other) to gain a complete picture of the results. This information can then be formatted for inclusion in the final report.

Q. 13. Explain Random number generation process.

Ans. In many modeling and simulation tools, random numbers are generated to model the timing and behavior of events. Simulators that predict the performance of a particular system may use Gaussian distributed random numbers to determine the probability of system failure. Monte-Carlo integrators use uniformly distributed random numbers scaled to a particular range to perform numeric integration. Exponential, Gaussian, or other distributed numbers may be used by simulators to estimate the change in time between two events. Random numbers have become an integral part of mathematical modeling techniques and are used throughout simulation applications. Since simulators often use millions of random numbers during their execution, the method used to generate these numbers is critical to the accuracy and speed of the simulator. This paper investigates the enhancement in simulator performance by generating random numbers with an FPGA. Using an FPGA, the simulator can run in parallel with the random number generation routine, significantly increasing its performance.

Q. 14. Write down the algorithm for Congurance Generator.

OR

Explain the linear congurential method. **[UTU: 2012-13]**

Ans. A Linear Congruential Generator (LCG) represents one of the oldest and best-known pseudo random generator algorithm. The theory behind them is easy to understand, and they are easily implemented and fast.

The generator is defined by the recurrence relation:

$$X_{n+1} \equiv (aX_n + c) \pmod{m}$$

where X_n 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 intiger constants that specify the generator. If $c = 0$, the generator is often called a multiplicative congruential method, or Lehmer RNG. If $c \neq 0$, the generator is called a mixed congruential method.

A pseudorandom number generator (PRNG), also known as a deterministic random bit generator (DRBG), is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state. Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom numbers are important in practice for simulations (e.g., of physical systems with the Monte Carlo method), and are central in the practice of cryptography and procedural generation.

Q. 15. Explain Uniformity Testing for Random number generation. **[UTU: 2012-13]**

Ans. Uniformity testing includes two tests named ad Kolmogorov-Smirnov and Chi- square tests.

Kolmogorov-Smirnov: The (K-S test) is a nonparametric test for the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K-S test), or to compare two samples (two-sample K-S test).

The Kolmogorov-Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. The null distribution of this statistic is calculated under the null hypothesis that the samples are drawn from the same distribution (in the two-sample case) or that the sample is drawn from the reference distribution (in the one-sample case). In each case, the distributions considered under the null hypothesis are continuous distributions but are otherwise unrestricted.

Chi-Suare Testing: A chi-square test (also chi squared test or X^2 test) is any statistical hypothesis test in which the sampling distribution of the test statistic is a chi-square distribution when the null hypothesis is true, or any in which this is asymptotically true, meaning that the sampling distribution (if the null hypothesis is true) can be made to approximate a chi-square distribution as closely as desired by making the sample size large enough.

Q. 16. Explain briefly the Independence Testing for random number generation.

[UTU: 2012-13]

Ans. In this case, an "observation" consists of the values of two outcomes and the null hypothesis is that the occurrence of these outcomes is statistically independent. Each observation is allocated to one cell of a two-dimensional array of cells (called a table) according to the values of the two outcomes. If there are r rows and c columns in the table, the "theoretical frequency" for a cell, given the hypothesis of independence, is

$$E_{ij} = \frac{\sum_k^c O_{i,k} \sum_k^r O_{k,j}}{N}$$

and fitting the model of "independence" reduces the number of degrees of freedom by $p = r + c - 1$. The value of the test-statistic, is

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{r} \frac{(O_{ij} - E_{i,j})^2}{E_{i,j}}$$

The number of degree of freedom is equal to the number of cells *rc*, minus the reduction in degrees of freedom, ρ, which reduces to $(r-1)(c-1)$.

For the test of independence, a chi-square probability of less than or equal to 0.05 (or the chi-square statistic being at or larger than the 0.05 critical point) is commonly interpreted by applied workers as justification for rejecting the null hypothesis that the row variable is independent of the column variable. The alternative hypothesis corresponds to the variables having an association or relationship where the structure of this relationship is not specified.

Q.17. Discuss process oriented programming. **[UTU:2012-13]**

Ans. Process-oriented programming is a programming paradigm that separates the concerns of data structures and the concurrent processes that act upon them. The data structures in this case are typically persistent, complex, and large scale - the subject of general purpose applications, as opposed to specialized processing of specialized data sets seen in high productivity applications (HPC). The model allows the creation of large scale applications that partially share common data sets. Programs are functionally decomposed into parallel processes that create and act upon logically shared data.

The paradigm was originally invented for parallel computers in the 1980s, especially computers built with transputer microprocessors by INMOS, or similar architectures. It evolved to meet deficiencies in the message passing paradigm of Occam and enable uniform efficiency when porting applications between distributed memory and shared memory parallel computers.

The first example of the paradigm appears in the programming language Ease designed at Yale University.

Q.18. Explain what you mean by Poisson process. Derive the Poisson distribution, given that the probability of single arrtival during a small time Dt is lDt and that of more than one arrival is negligible. **[UTU: 2011-12]**

Ans. Derivations of the formulae's for Position process: A Poisson process is a particular random process in time. Associated with it are two distributions. The first is the Poisson distribution $P_n(t)$, which is the probability that precisely n events happen from time 0 to time t. It is a pmf in n, for fixed value of time t. The second is the exponential distribution $P(t)$, which is a pdf in t. $P(t)$ is the pdf for an event happening after a particular time t. It is also the pdf for the inter-event time for events.

Let an event E have constant probability to occur per time, i.e., a Poisson process. Let's say the probability for it to occur in a small time interval dt is given by λdt. The Poisson distribution $P_n(t)$ is the probability that after the elapse of time t, n events happened. This is a p.m.f. in n, but not a p.d.f. in t.

$$P_n(t) = e^{-\lambda t}(\lambda t)^n/n!$$

In particular,

$$P_0(t) = e^{-\lambda t}.$$

$P_0(t)$ is the chance that no events have taken place at time t, starting from $t = 0$. If we now subdivide the interval $[0, t]$ in small segments of size dt, the event E should not happen in each of those intervals. The chance for no event in a single interval is $1 - \lambda t$. Therefore the chance it does not happen in any interval is

$$P_0(t) = (1 - \lambda dt)^N$$

with $N = t/dt$ the number of intervals. In the limit $dt \to 0$ we have, using a well-known formula for the exponential.

$$P_0(t) = \lim_{dt \to 0} (1 - \lambda dt)^{(t/dt)} = e^{-\lambda t}.$$

What is now the p.d.f. $P(t)$ for the event to happen at a particular time t? (Note this is a continuous distribution with $\int_{-\infty}^{\infty} P(t)\,dt = 1$. $P(t)\,dt$ is the probability the event will happen in the infinitesimal time interval $[tt + dt]$. We know the chance for the event to happen in an interval dt, namely λdt. For the event to happen in $[tt + dt]$ it should *not* happen until time t and happen in the next time interval of size dt. The chance for is *not* to happen until time t is just $P_0(t) = e^{-\lambda t}$. The product of the two is $e^{-\lambda t}\lambda dt$, from which it follows that

$$P(t) = \lambda e^{-\lambda t}.$$

$P(t)$ is called the exponential distribution. The inter-event time is distributed according to $P(t)$.

What remains to be derived is the rest of the Poisson distribution $P_k(t)$ for $k > 0$. Let us derive a formula for $P_n(t + dt)$. $P_n(t + dt)$ is the chance for n events to happen at time $t + dt$. This can happen by n events happening until time t and no events in the interval $[tt + dt]$, or by $n - 1$ events to happen until time t and one event in the interval $[tt + dt]$. This means

$$P_n(t + dt) = P_n(t)(1 - \lambda dt) + P_{n-1}(t)\lambda dt.$$

Substituting

$$P_n(t + dt) = P_n(t) + dt\frac{dP_n(t)}{dt}$$

gives $$\frac{dP_n(t)}{dt}dt = \lambda(-P_n(t) + P_{n-1}(t))dt$$

Dividing out dt gives

$$\frac{dP_n(t)}{dt} = \lambda(P_{n-1}(t) - P_n(t)).$$

This is a recursive differential equation for $P_n(t)$. It can seen easily by substitution the formula for the Poisson distribution that $P_n(t) = e^{-\lambda t}(\lambda t)^n/n!$ indeed satisfies this equation.

Q.19. Show that if the inter-arrival times are negative exponentially distributed the number of arrivals in a time period is Poisson process and conversely. [UTU: 2011-2012]

Ans. A common situation is that the arrivals are said to be completely random. This means that an arrival can occur at any time. Subject only to the restriction that the mean arrival rate be some given value. It is assumed that the time of the next arrival is independent of the previous arrival, that the probability of an arrival in an interval Δt is proportional to Δt. In fact λ is the mean number of arrivals per unit time then the probability of an arrival in Δt is $\lambda\Delta t$. With these assumptions, it is possible to show that the distribution of inter-arrival times is exponential. The probability density function of inter-arrival time is given by

$$P(t) = \lambda e^{-\lambda t} \ (t \geq 0)$$

It follows arrival distribution is

$$A_0(t) = e^{-\lambda t}$$

It can be shown that with an exponential distribution of inter-arrival times, the probability of n arrivals occurring in a period of length t is given by

$$P(n) = \frac{(\lambda t)^n e^{-t}}{n!} (n = 0, 1, 2, ...)$$

This distribution, which is called the Poisson distribution, is discrete.

Q.20. Explain a system with Poisson input, exponential waiting time with single channel. Also determine the average length of the waiting time. [UTU: 2011-12]

Ans. It is queuing model where the arrival follows a Poisson process, Service times are exponentially distributed and there is only one server. It is a system with Poisson input, exponential waiting time & Poisson output with single channel.

A probability of zero unit in the queue

$$P_0 = 1 - \frac{\lambda}{\mu}$$

Average queue length $(L_q) = \frac{\lambda^2}{\mu(\mu - \lambda)}$

Q. 21. How do programmers deal with exceptional situations is simulation process? [UTU: 2011-12]

Ans. When an exceptional situation arises in a program, there are a number of possible actions that can be taken.

Ignoring Exceptional Situations: Our description of exceptional situations is general enough to include situations which are not normal but which the program does not recognize as such. The example of how compiled calls to + behave in **MACLISP** is an example of this behaviour. Hence, one possible way in which programs can deal with exceptional situations is to fail to recognize them.

While such failure might not be something to encourage as a programming style, it is important to recognize that is does occur - frequently, in fact. It can happen any time an assumption of any kind is made by code but not mechanically verified (either at compile time or runtime).

Modified Return Value Conventions: In some situations, the type of the return value is highly constrained. For example, since we know that many operations on numbers yield only other numbers, we could use non-numeric return value from such functions to convey other kinds of information. Our earlier suggestion that **PLUS** might return *false* when passed non-numeric arguments is an example of this technique.

Non-local Transfer of control: Sometimes, a convention may have been established between a group of programs so that when certain exceptional situations are detected, control is transferred to one of the callers in a non local fashion. Such a transfer is generally accomplished by an escape procedure in SCHEME or the THROW primitive in Lisp.

Also in this category are the mechanisms some languages provide for aborting and restarting computations. These are typically just "syntactic sugar" for the same kind of non-local transfer of control already provided by THROW. For example, the tG function (which simulates the effect of the Maclisp abort character, Control-G) "throws" to the point which restarts the toplevel Read-Eval-Point loop.

Q.22. For *N* | *M* |1 queuing system, find expected value of queue length and probability distribution of waiting time. [UTU: 2011-12]

Ans. Average queue length $= \frac{P}{1-P} - P = \frac{P^2}{1-P}$

The probability of n customers being in the system can also be expected as

$$P_n = P^n(1-P)$$

Q.23. The sequence of numbers 0.54, 0.73, 0.98, 0.11 and 0.68 has been generated. Use the Kokmogorov-Smirnov test with $\alpha = 0.05$ to determine if the hypothesis that the number are uniformly distributed on the interval [0, 1] can be rejected. [UTU: 2012-13]

Ans. Given numbers are 0.54, 0.73, 0.98, 0.11 and 0.68. Here N = 5.

First the number must to ranked from smallest to largest.

Prepare the table forKolmogorov-Smirnov test as given below:

i	1	2	3	4	5
$R_{(i)}$	0.11	0.54	0.68	0.73	0.98
$i/N - R_{(i)}$	0.09	–	–	0.07	0.02
$R_{(i)} - (i-1)/N$	0.11	0.34	0.28	0.13	0.18

The second row from top lists the numbers from smallest ($R_{(1)}$) to largest ($R_{(5)}$). The computations for D^+, namely $i/N - R_{(i)}$ and for D^-, namely $R_{(i)} - (i-1)/N$, are performed. The statistics are computed as $D^+ = 0.09$ and $D^- = 0.34$.

Therefore, $D = \max\{0.09, 0.34\} = 0.34$.

The critical value of D for $a = 0.05$ and $N = 5$ is 0.565 (given) since the computed value, 0.34 is less than the tabulated critical value, 0.565, the hypothesis of no difference between the distribution of the generated numbers and the uniform distribution is not rejected.

❑❑❑

UNIT 3

RANDOM VARIATE GENERATION

Q. 1. Write a short note on Random Variate Generation.

Ans. Random variate generation (RVG) is a vital part of the domain of any discrete event simulation study. It comes into play whenever there is a need to simulate the uncertainty in the conduct of an entity in the system under study. When the sample of randomness of the entity to be analyzed is the identified the operation of entity is said to follow a particular stochastic distribution.

Any random process is described by:

1. Gathering of data on random phenomenon.
2. Approximating the acquired data to a known probability distribution.
3. Inference of parameter of the probability distribution.

A variable that is taken from an identical distribution of pseudo random number is called a RANDOM VARIATE. Stochastic models, while simulating are often referred to by Random Variate. Random variate may passes uniform distribution or non-uniform distribution.In general Random variate refers to a particular value of random variable. The process of random variate generation refer to generation of random variates for a given Random variable. this method that is responsible for generating Random variate is called RANDOM VARIATE GENERATOR.

There are two types of Random Variate Generator:

1. **Uni-Variate:** It involves the generation of single variate at a time.
2. **Multi-Variate:** It involves the generation of a vector of variate at a time, which do not show mutual independence.

Q. 2. Explain Random variate Parameters. Briefly

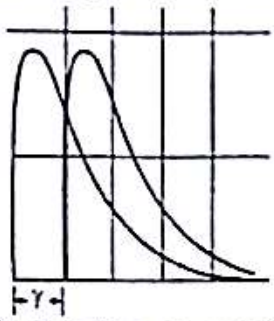
Ans. Distributions can have any number of parameters. Do note that as the number of parameters increases, so does the amount of data required for a proper fit. In general, most distributions used for reliability and life data analysis, the lifetime distributions, usually are limited to a maximum of three parameters. These three parameters are usually known as the scale parameter, the shape parameter and the location parameter.

Scale parameter: The scale parameter is the most common type of parameter. The majority of distributions in reliability or survival analysis field have a scale parameter. In the case of one-parameter distributions, the sole parameter is the scale parameter. The scale parameter defines where the bulk of the distribution lies, or how stretched out the distribution is. In the case of the normal distribution, the scale parameter is the standard deviation. scale parameter is denoted by "n"

Shape parameter: The shape parameter, as the name implies, helps define the shape of a distribution. Some distributions, such as the exponential or normal, do not have a shape parameter since they have a predefined shape that does not change. In the case of the normal distribution, the shape is always the familiar bell shape. The effect of the shape parameter on a distribution is reflected in the shapes of the reliability function and the failure rate function. shape parameter is denoted by "β"

Location parameter: The location parameter is used to shift a distribution in one direction or another. The location parameter, usually denoted as, defines the location of the origin of a

distribution and can be either positive or negative. In terms of lifetime distributions, the location parameter represents a time shift. Location parameter is denoted by "γ".



This means that the inclusion of a location parameter for a distribution whose domain is normally [0, ∞] will change the domain to [γ, ∞], where ?can be either positive or negative. For a positive location parameter, this indicates that the reliability for that particular distribution is always 100% up to that point γ. In other words, a failure cannot occur before this time γ.

2. 3. Explain briefly the Continous Probability Distribution.

Ans. A continuous probability distribution shall be understood as a probability distribution that has probability density function. Mathematicians also call such distribution absolutely continuous, since its cumulative distribution function is absolutely continuous with respect to the Lebesgue measure γ. If the distribution of X is continuous, then X is called a continuous random variable. There are many examples of continuous probability distributions: normal, uniform, chi-squared, and others.

Intuitively, a continuous random variable is the one which can take a continuous range of values - as opposed to a discrete distribution, where the set of possible values for the random variable is at most countable. While for a discrete distribution an event with probability zero is impossible (e.g. rolling 3½ on a standard die is impossible, and has probability zero), this is not so in the case of a continuous random variable. For example, if one measures the width of an oak leaf, the result of 3½ cm is possible, however it has probability zero because there are infinitely many other potential values even between 3 cm and 4 cm. Each of these individual outcomes has probability zero, yet the probability that the outcome will fall into the interval (3 cm, 4 cm) is nonzero. This apparent paradox is resolved by the fact that the probability that X attains some value within an infinite set, such as an interval, cannot be found by naively adding the probabilities for individual values. Formally, each value has an infinitesimally small probability, which statistically is equivalent to zero.

Formally, if X is a continuous random variable, then it has a probability density function $f(x)$, and therefore its probability to fall into a given interval, say [a, b] is given by the integral

$$Pr[a \le X \le b] = \int_a^b f(x)\,dx$$

In particular, the probability for X to take any single value a (that is $a \le X \le a$) is zero, because an integral with coinciding upper and lower limits is always equal to zero.

The definition states that a continuous probability distribution must possess a density, or equivalently, its cumulative distribution function be absolutely continuous. This requirement is stronger than simple continuity of the cdf, and there is a special class of distributions, singular distributions, which are neither continuous nor discrete nor their mixture. An example is given by the Cantor distribution. Such singular distributions however are never encountered in practice.

Note on terminology: some authors use the term "continuous distribution" to denote the distribution with continuous cdf. Thus, their definition includes both the (absolutely) continuous and singular distributions.

By one convention, a probability distribution μ is called continuous if its cumulative distribution function $F(x) = \mu(-\infty, x)$ is continuous and, therefore, the probability measure of singletons $\mu\{x\} = 0$ for all x.

Another convention reserves the term continuous probability distribution for absolutely continuous distributions. These distributions can be characterized by a probability density function: a non-negative Lebesgue integrable function f defined on the real numbers such that

$$F(x) = \int_{-\infty}^{x} f(t)\, dt$$

Q .4. Explain Discrete Probability Distribution for random variate generation.

Ans. A discrete probability distribution shall be understood as a probability distribution characterized by a probability mass function. Thus, the distribution of a random variable X is discrete and X is then called discrete random variable.

$$\sum_{u} \Pr(X = u) = 1$$

as u runs through the set of all possible values of X. It follows that such a random variable can assume only a finite or countable infinite number of values.

In cases more frequently considered, this set of possible values is a topologically discrete set in the sense that all its points are isolated points. But there are discrete random variables for which this countable set is dense on the real line (for example, a distribution over rational number).

Among the most well-known discrete probability distributions that are used for statistical modeling are :

1. Poisson Distribution.
2. Bernoulli Distribution.
3. Binomial Distribution.
4. Geometric Distribution.
5. Negative Binomial Distribution.

In addition the Discrete Uniform Distribution is commonly used in computer programming that make equal probability random selection between a number of choices.

Q. 5. Explain the Inverse-Transform Method of producing random variate for exponential distribution with example.

OR

Suggest a step by step procedure to generate random variates using inverse transform technique for exponential distribution. **[UTU: 2012-13]**

Ans. The inverse-transform technique can be used to sample from the exponential, the uniform, the weibull, the triangular distribution and from empirical distribution. Additionally, it is the underlying principle for sampling from wide variety of discrete distribution. Computationally, it is the most straightforward, but not always the most efficient, technique. The inverse-transform method can be utilized, at least in principle, for any distribution, but it is most useful when the cdf, F(x), is of a form so simple that its inverse, F-1, can be computed easily.

Inverse-transform method consist of the following steps:

1. Compute the cdf of the desired random variable X.
2. Set F(X) = R on the range of X.
3. Solve the equation F(X) = R for X in terms of R.

4. Generate (as needed) uniform random number R_1, R_2, R_3, R_4,.....and compute the desired random variate by:

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

Example: Suppose we wish to generate random numbers according to N(0; 1), using our source of uniform random numbers, u, on (0; 1). Our prescription says to solve for x in:

$$u = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-x'^2}{2}} - dx'$$

This looks hard, but there is a trick { consider a two-dimensional distribution, and transform to polar coordinates:

$$p(x, y)dxdy = f(x)\, f(y)\, dxdy$$

$$= re^{\frac{r^2}{2}}\, dr\, \frac{d\phi}{2\pi}$$

$$= g(r)\ h(\phi) drd\ \phi$$

Now generate r and _ according to $g(r)$ and $h(\phi)$, and transform the result to x and y obtaining two normally-distributed random numbers.

That is, sample two uniform random numbers u and v on (0; 1) and let

$$u = \int_{1}^{\infty} re^{0\frac{r^2}{2}}\, dr = e^{-\sqrt{2r}}$$

where we have used for convenience

$$u = 1 - \int_{0}^{\infty} [(f)]x')dx'$$

Thus, $r = \sqrt{-2 \mid nu_2}$. Similarly, we have

$$\phi = \upsilon * 2\pi$$

Then $x = r \cos \theta$ and $y = r \sin \theta$ are each independent samplings from the desired $N(0; 1)$ distribution.

Q. 6. What is Composition Method? Explain.

Ans. The two dominant overheads, i.e. event translation and scheduling overheads, are closely related to the mechanism of passing the four types of messages. They can be significantly reduced by applying a kind of compiled simulation technique, called composition. Composition is a process of merging all the component models belonging to a coupled model. As the composed model is also an atomic model, we can get a finally composed model of the form of an atomic model after applying a series of composition.

If the distribution we wish to generate is in the form of a sum of terms, it may be easier, or save computation time, to break it up into pieces. For example, suppose we can decompose the desired PDF (probability density function) f(x) as:

$$f(x) = rf_a(x) + (1 \mid r)f_b(x);$$

where f_a and f_b are themselves normalized PDFs $(0 \le r \le 1)$ provides an illustration of such a decomposition. Then we can generate a sampling from f by:

1. Generate two uniform randoms u_1 and u_2.
2. If $u_1 < r$ let $x = F_a^{-1}(u_2)$
   If $u_1 \le r$ let $x = F_b^{-1}(u_2)$.

Q. 7. Explain briefly the Acceptance-Rejection Method for random variate generation.

[UTU: 2012-13]

Ans. It may be possible that computing the inverse transform, even on a decomposed distribution, is intractable. A powerful (but potentially inefficient) method is the\acceptance-rejection method". This method can be used even if we can't readily compute the normalization of f(x). That is, we may only know the functional form $Af(x)$, where A is unknown.

Here is the algorithm:

Step1: Generate a random number R.

Step2a: If R ≥ 1/4, accept $X = R$, then go to step 3.

Step2b: If R ≥ 1/4, reject R,return to step 1.

Step3: If another uniform random variate on [1/4] is needed, repeat the procedure beginning at step 1. If not stop.

Each time step 1 is executed, anew random number R must be generated. step 2a is an "acceptance" and step2b is a "rejection" in this acceptance- rejection method. To summarize the technique, random variates (R) with some distribution (here uniform on [0,1] are generated until some condition (R > 1/4) is satisfied. When the condition is finally satisfied, the desired random variate, X (here uniform on [1/4,1]), can be computed (X = R).

This procedure can be shown to be correct by recognizing that the accepted values of R are conditioned values : that is, R itself does not have the desired distribution, but R conditioned on the event {R?1/4} does have the desired distribution. To show this, take $1/4 \le a < b \le 1$; then

$$p(a < R \le b \mid 1/4 \le R \le 1) = \frac{p(a < R \le b)}{p\left(\frac{1}{4} \le R \le 1\right)} = \frac{b-a}{\frac{3}{4}}$$

Which is correctly probability for a uniform distribution.

The efficiency of acceptance- rejection method depends heavily on being able to minimize the number of rejection.Here the probability of "rejection" is P(R<1/4)=1/4, and the probability of "success" is 3/4.

Acceptance-rejection method or an alternative procedure is the more efficient depends several considerations. The computer being used, the skills of the programmer and the relative inefficiency of generating the additional random number needed by acceptance-rejection should be compared to the computation required by the alternative procedure.

Q.8. Explain Chi-square goodness of fit test for exponential distribution, with an example.

[UTU: 2012-13]

Ans. Being the most often used the classical Pearson Chi-square test is a convenient test if a broad class of alternatives is of interest. Thisχ^2-text is easily computed and has approximately the χ^2_{z-1} distribution for rather small sample sizes, z being the number of grouping intervals. In 1928 R.A. Fisher proposed to use a Chi-square statistic to verify the hypothesis that the distribution function belongs to the family of continuous function depending on unknown parameters. He showed that the limiting distribution of Pearson's statistic χ^2 is in this case not χ^2_{z-1} and that its distribution depends on how unknown parameters are estimated. He also showed that Pearson's statistics will have the χ^2_{z-s-1} limiting distribution (s is the number of parameters to be estimated) only if one estimates unknown parameters by minimum Chi-square estimators based on grouped data.

In the 1950's Chernoff, Lehmann (1954) and Watson (1958) showed that a formal using of Person's test estimating unknown parameters by ungrouped data leads to a substantial difference of a limiting distribution from the Chi-square distribution even if one groups data correctly. This

limiting distribution generally turns out to be dependent on unknown parameters making it impossible to tabulate quantities.

Example: A new casino game involves rolling 3 dice. The winnings are directly proportional to the total number of sixes rolled. Suppose a gambler plays the game 100 times, with the following observed counts:

Number of Sixes	Number of Rolls
0	48
1	35
2	15
3	3

The casino becomes suspicious of the gambler and wishes to determine whether the dice are fair. What do they conclude?

If a die is fair, we would expect the probability of rolling a 6 on any given toss to be 1/6. Assuming the 3 dice are independent (the roll of one die should not affect the roll of the others), we might assume that the number of sixes in three rolls is distributed Binomial(3, 1/6). To determine whether the gambler's dice are fair, we may compare his results with the results expected under this distribution. The expected values for 0, 1, 2, and 3 sixes under the Binomial(3, 1/6) distribution are the following:

Null Hypothesis:

p_1 = P(roll 0 sixes) = P(X = 0) = 0.58

p_2 = P(roll 1 six) = P(X = 1) = 0.345

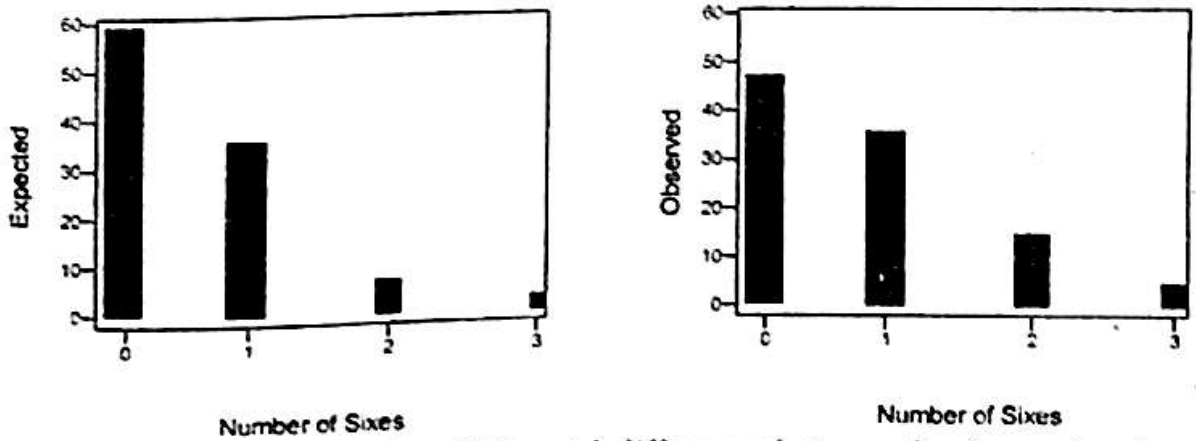
p_3 = P(roll 2 sixes) = P(X = 2) = 0.07

p_4 = P(roll 3 sixes) = P(X = 3) = 0.005.

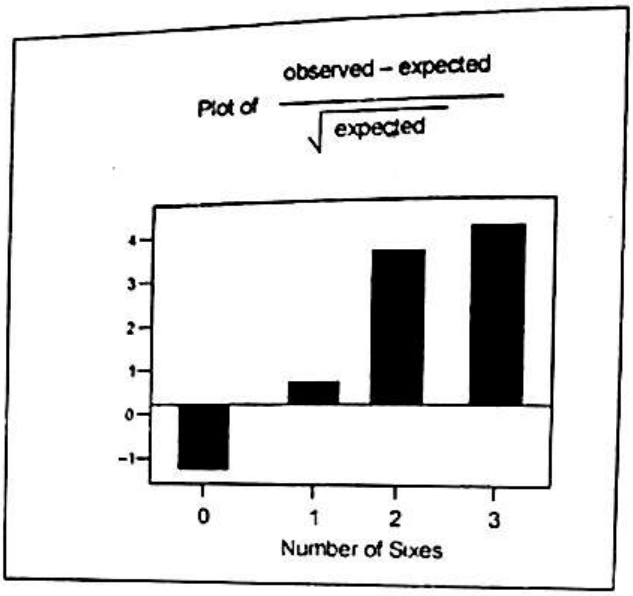
Since the gambler plays 100 times, the expected counts are the following:

Number of Sixes	Expected Counts	Observed Counts
0	58	48
1	34.5	35
2	7	15
3	0.5	3

The two plots shown below provide a visual comparison of the expected and observed values:



From these graph, it is difficult to distinguish differences between the observed and expected counts. A visual representation of the differences is the chi-gram, which plots the observed-expected counts divided by thesquare root of the expected counts, as shown below:



The chi-square statistic is the sum of the squares of the plotted values.
$(48\text{-}58)^2/58 + (35\text{-}34.5)^2/58 + (15\text{-}7)^2/7 + (3\text{-}0.5)^2/0.5 = 1.72 + 0.007 + 9.14 + 12.5 = 23.367.$

Q.9. Discuss about random number. Write code for Poisson distribution function.

[UTU: 2011-12]

Ans. In experimental work e.g. in physic one often encounters problem where a standard statistical probability density function is applicable. It is often of great help to be able to handle these in different ways such as calculating probability contents or generating random numbers. In modern computing Monte Carlo simulations are of vital importance and we give methods to achieve random numbers from the distributions.

Random number can be generated in various ways:

- **Uniform distribution**
  "Pick a number randomly in the interval [0, 1]"
  `rand(1,1); rand(n, m);`
- **Zero-one distribution**
  "Generate sequence of m random digital numbers, 1 with prob p, 0 with prob 1-p"
  `(rand(1,m) <=p);`
- **Binomial distribution simbinom.m**
  "Perform n trials where the success probability each time is *p*, let *x* be the number of successes"
  `x=sum(rand(n,m)<=p); %x is vector of m random numbers from Bin(n,p)`
- **Normal distribution**
  Gaussian, or normally distributed, random numbers are generated using the Matlab command RANDN.

```
randn(1,m);                 % vector of length m of random numbers from N(0, 1)
randn(n, m);                % n × m-matrix with N(0,1) entries
Mu+sigma.*randn(1,m); % m numbers from the N(mu, sigma^2)-distirbution
```

- **Exponential distribution simexp.m**

"An event occurs randomly in time with *intensity* lambda, i.e. the probability the event occurs between time t and $t + h$ is lambda h, for small h, for small h. Let t be the waiting time until the event occurs."

```
t=-log(rand)/lambda;          % return number from Exp (lambda)
t=-log(rand(1,m)./lambda; % m random numbers from Exp (lambda)
```

- **Geometric distribution simgeom.m**

"Perform trials where the success probability each time is p, let x be the number of failures before the first success".

```
% a vector of m random number from Ge(p)
  x=floor(-log(rand(1, m)./(-log(1 - p)));
```

- **Poisson distribution**

Matlab Statistics Toolbox has built in command POISSRND for generating random numbers from the Poisson distribution.

```
poissrnd(lambda);
%N×M-matrix of random numbers from P0(lambda)
poissrnd (lambda, n, m);
```

If not available use e.g.

```
arrival=cumsum(-log(rand(1.5)./lambda)))'
n-length (find(arrival<=lambda));'
```

Poisson Distribution

In *probability theory* and *statis* a *discrete probability distribution* that expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and *independently* of the time since the last event. (The Poisson distribution can also be used for the number of events in other specified intervals such as distance, area of volume.) stics, the **Poisson distribution.**

If the *expected number* of occurrences in a given interval is λ, then the probability that there are exactly k occurrences (k being a non-negative *integer*, $k = 0, 1, 2, ...$) is equal to

$$f(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Generating Poisson-distributed random variables

A simple algorithm to generate random Poisson-distributed numbers (Pseudo-random number sampling) has been given by *Knuth:*

```
algorith poisson random number (knuth);
 init:
     Let L ← e^-λ, k ← 0 and p ← 1.
 do
     k ← k + 1.
     Generate uniform random number u in [0, 1] and let p ← p × u.
 while p > L.
 return k - 1.
```

While simple, the complexity is linear in λ. There are many other algorithms to overcome this. Some are given in Ahrens and Dieter. Also, for large values of λ, there amy be numerical stability issues because of the term $e^{-\lambda}$. One solution for large values of λ is *Rejection sampling*, another is to use a Gaussian approximation to the Poisson.

Parameter estimation

Maximum likelihood: Given a simple of n measured values k_i we wish to estimate the value of the parameter λ of the Poisson population from which the sample was drawn. To calculate the *maximum likelihood* value, we form the log-likelihood function

$$L(\lambda) = \ln \prod_{i=1}^{n} f(k_i|\lambda) = \sum_{i=1}^{n} \ln\left(\frac{e^{-\lambda}\lambda^{k_i}}{k_i!}\right)$$

$$= -n\lambda + \left(\sum_{i=1}^{n} k_i\right)\ln(\lambda) - \sum_{i=1}^{n} \ln(k_i!)$$

Take the derivative of L with respect to λ and equate it to zero:

$$\frac{d}{d\lambda}L(\lambda) = 0 \quad \Leftrightarrow \quad -n + \left(\sum_{i=1}^{n} k_i\right)\frac{1}{\lambda} = 0$$

Solving for λ yields a stationary point, which if the second derivative is negative is the maximum-likelihood estimate of λ.

$$\hat{\lambda}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} k_i$$

Checking the second derivative, it is found that it is negative for all λ and k_i greater than zero, therefore this stationary point is indeed a maximum of the initial likelihood function:

$$\frac{\partial^2 L}{\partial \lambda^2} = -\lambda^{-2}\sum_{i=1}^{n} k_i$$

Since each observation has expectation λ so does this sample mean. Therefore it is an *unbiased estimator* of λ. It is also an efficient estimator, i.e., its estimation variance achieves the *Cramer-Rao lower bound* (CRLB). Hence it is *MVUE*. Also it can be proved that the sample mean is complete and sufficient statistic for λ.

Q.10. Write a program to generate uniformly distributed random numbers between 0 and 1 (with as many significant digits as your computer word can hold). Study and comment on the behaviour of the least significant digit of these numbers generated. **[UTU: 2011-12]**

Ans. The following FORTRAN Function is used for generating uniformly distributed pseudorandom numbers in he range (0, 1) on the IBM 7044 computer - a 36-bit binary machine.

```
Function RNDYI(DUM)
Integer A, X
DATA A, X/189277, 11750920161/
X=A*X
AX = X
RNDYI=AX/34359738368.0
```

```
RETURN
END
```

The user must, however, put some dummy argument (DUM) just to comply with the FORTRAN requirement for using and defining function.

It should be noted that the least significant digits of pseudo-random number generated by the multiplicative congruential method are not random. Therefore low-order digits should not be used as random number.

Random number generators are used so frequently that almost every computer manufacture provides one or more sub-programs tailored to this particular system.

Q.11. A company hiring out building equipment has six mechanical diggers that are in thigh demand. Customers hire the digger for three days at a time. Assume that customers arrive with an exponentially distributed inter-arrival time of mean six hours. If a customer arrives and all the diggers are on hire, he takes his customer to a competitor. Simulate this system for a month and estimated the percentage of lost customer. Assume that the company is open for business 24 hours a day. **[UTU: 2011-12]**

Ans. Company hires diggers = 6

Customer hires a digger form 3 days

Customer arrives at mean 6 hours

If the company is open for business 24 hrs a day.

⇒ On a day, 4 customers arrived

⇒ On First day, all 4 customers got digger

⇒ On the second day, two customers are lost

⇒ On the third day, all four customers are lost

⇒ On the fourth day, all four customers got digger.

⇒ On the fifth day, two customers are lost

⇒ On the sixth day, again all four customers are lost.

It means 6 customers are lost in 3 days.

That is 2 customers are lost per day.

⇒ 50 % of the customer in a day is lost.

❑❑❑

UNIT 4

QUEUING MODELS

Q. 1. What are Queuing Models in Simulation? Explain.

OR

Explain characteristics of queuing models. List different queuing notations. [UTU: 2012-13]

Ans. In queuing theory,a queuing model is used to approximate a real queueing situation or system, so the queuing behaviour can be analysed mathematically. Queuing models allow a number of steady state performance measure to be determined, including:

- The average number in the queue, or the system,
- The average time spent in the queue, or the system,
- The statistical distribution of those numbers or times,
- The probability the queue is full, or empty, and
- The probability of finding the system in a particular state.

These performance measures are important as issues or problems caused by queueing situations are often related to customer dissatisfaction with service or may be the root cause of economic losses in a business. Analysis of the relevant queueing models allows the cause of queueing issues to be identified and the impact of proposed changes to be assessed.

Queuing models can be represented using kendall's notation:

A/B/S/K/N/D

where:

- A is the inter arrival-time distribution
- B is the service time distribution
- S is the number of servers
- K is the system capacity
- N is the calling population
- D is the service discipline assumed

Many times the last members are omitted, so the notation becomes A/B/S and it is assumed that $K = \infty$, $N = \infty$ and D = FIFO.

Some standard notation for distributions (A or B) are:

- M for a Markovian (poisson, exponential) distribution
- E_κ for an Erlang distribution with κ phases
- D for degenerate (or deterministic) distribution (constant)
- G for general distribution (arbitrary)
- PH for a phase-type distribution.

Q. 2. Define and Proove Little's Theorem. **[UTU: 2012-13]**

Ans. Little's Theorem (sometimes called Little's Law) is a statement of what was a "folk theorem" in operations research for many years:

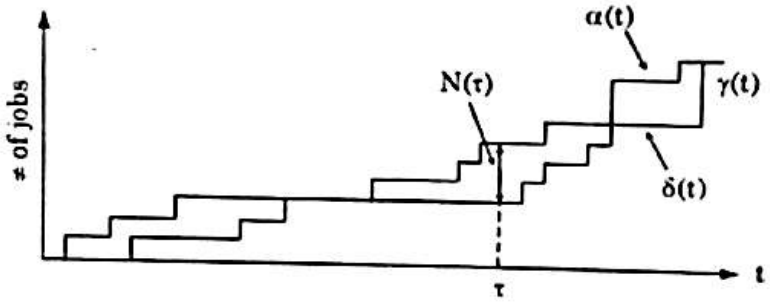
$$\bar{N} = \lambda \bar{T} \qquad ...(1)$$

Where N is the random variable for the number of jobs or customers in a system, λ is the arrival rate at which jobs arrive, and T is the random variable for the time a job spends in the system (all of this assuming steady-state). What is remarkable about Little's Theorem is that it applies to any system, regardless of the arrival time process or what the "system" looks like inside.

PROOF: Define the following;

- $a(t)$ = number of arrival in the interval $[0, t]$.
- $\delta(t)$ = number of departure in the interval $[0, t]$.
- $N(t)$= number of jobs in the system at time $a(t) - \delta(t)$.
- $\gamma(t)$ = accumulated customer - seconds in $[0, 1]$.

These functions are graphically shown in the following figure:



The shaded area between the arrival and departure curves is $\gamma(t)$.

$\lambda(t)$ = arrival rate over interval $[0, t] = a(t)/t$.

$\bar{N}_t$ = average number of jobs during the interval $[0,t] = \gamma(t)/t$.

$\bar{T}_t$ = average time a job spends in the system in $[0,t] = \dfrac{\gamma(t)}{a(t)}$.

$$\Rightarrow \qquad \gamma(t) = T_t a(t)$$

$$\Rightarrow \qquad \bar{N}_t = \frac{\bar{T}_t\,(a(t))}{t}\,\lambda(t) T_t$$

Assume that the following limit exist:

$$\lim_{\tau \to \infty} \lambda(t) = \lambda$$

and $$\lim_{\tau \to \infty} \bar{T}_\tau = \bar{T}$$

Then $$\lim_{\tau \to \infty} \bar{T}_\tau = \bar{T}$$

Also exist and is given by = $\bar{N} = \lambda \bar{T}$ (proved)

Q. 3. Give the application or use of little's theorem.

Ans. Application of little's theorem are :

- **Estimate Waiting Times:** [W = Average Number of Customers / Average Throughput] (as the patient flow example above)
- **Planned Inventory Time:** Suppose a product is scheduled so that we expect it to wait for 2 days in finished goods inventory before shipping to the customer. This two days is called planned inventory time and is sometimes used as protection against system variability to ensure high delivery service. Using Little's Law, the total amount of inventory in finished goods can be computed as [FGI = Throughput x Planned Inventory Time]
- **WIP Reduction:** Reducing WIP in a process without making any other changes will also reduce throughput. So, simply reducing inventory is not enough to achieve a Lean Manufacturing system. An integral part of any Lean Manufacturing implementation is an effort to reduce variability (often the domain of Six Sigma), to enable a line to achieve the same (or greater) throughput with less WIP.
- Little's Law for Product Development

Q. 4. What do you mean by M/M/1 model? Explain its analytical result in detail?

[UTU: 2012-13]

Ans. The M/M/1 is a single-server queue model, that can be used to approximate simple systems.

The average arrival and service rates are not enough to allow us to compute the average time an entity waits for service in a queueing system. To do this, we must introduce variability into arrival and service processes. The most straightforward queueing model that incorporates variability is the single server system with Markov (memory less) arrival and service processes.

We call this the M/M/1 queue, where the first M indicates that the arrival process is Markov, the second M indicates that the service process is Markov and the 1 indicates that there is only a single server. Waiting space is assumed infinite, so there is no limit on the length of the queue.

Analytical Result: Such a system can be modelled by a birth-death process, where each state represents the number of users in the system. As the system has an infinite queue and the population is unlimited, the number of states the system can occupy is infinite: state 0 (no users in the system), state 1 (1 user), state 2 (two users), etc. As the queue will never be full and the population size being infinite, the birth rate (arrival rate), λ, is constant for every state. The death rate (service rate), μ, is also constant for all states (apart from in state 0). In fact, regardless of the state, we can have only two events:

- A new user arrives. So if the system is in state k, it goes to state $k + 1$ with rate λ.
- A user leaves the system. So if the system is in state k, it goes to state $k - 1$ (or k if k is 0)

It's easy now to see that the system is stable only if $\lambda < \mu$. In fact if the death rate is less than the birth rate, the average number of users in the queue will become infinite. i.e. the system will not have an equilibrium.

The model can reveal interesting performance measures of the system being modelled, for example:

- The mean time a user spends in the system
- The mean time a user spends waiting in the queue
- The expected number of users in the system
- The expected number of users in the queue
- The throughput (Number of users served per unit time).

We can define

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

The probability the system is in state i can be easily calculated:

$$\text{Prob}\,(q = i) = \pi_i = (1 - \rho)\rho^i$$

With this information, the performance measures of interest can be found; for example:

- The expected number of users in the system N is given by

$$\bar{N} = \frac{\rho}{1 - \rho}, \text{ and its variance by}$$

$$\sigma_N^2 = \frac{\rho}{(1 - \rho)^2}$$

- The expected number of requests in the server

$$\bar{N}_S = \lambda \bar{x} = \rho \text{ and so on.....}$$

Q. 5. Explain the Birth Date Process model in detail.

Ans. Birth Date Process model: The average arrival and service rates are not enough to allow us to compute the average time an entity waits for service in a queuing system. To do this, we must introduce variability into arrival and service processes. The most straightforward queuing model that incorporates variability is the single server system with Markov (memory less) arrival and service processes. We call this the M/M/1 queue, where the first M indicates that the arrival process is Markov, the second M indicates that the service process is Markov and the 1 indicates that there is only a single server. Waiting space is assumed infinite, so there is no limit on the length of the queue.

The assumption that both the arrival and service processes are memory less means that the only piece of information we need to predict the future evolution of the system is the number of entities in the queue. For example, in the Macro hard support center if there is currently one customer being served and one waiting on hold, then we know that the expected time until the customer completes service is 15 minutes and the expected time until the next customer calls for service is 20 minutes. Because both inter arrival and service times are exponentially distributed, and hence memory less, we don't need to know how long the current customer has been in service or how long it has been since a call came into the system.

This fact allows us to define the number of customers in the queue as the system state.

Moreover, since arrivals come into the system one at a time and customers are also serviced one at a time, the state of the system either increases (when a customer arrives) or decreases (when a service is completed) in increments of one. Systems with this property are called birth-death processes because we can view increases in the state as "births" and decreases as "deaths".

We can use the birth-death representation of the M/M/1 queue to compute the long run average probabilities of finding the system in any given state. We do this by noting that in steady state, the average number of transitions from state i to state i+1 must equal the average number of transitions from state i+1 to state i. Graphically, the number of times the birth-death process in Figure 4 crosses the vertical line going left to right must be the same as the number of times it crosses the vertical line going right to left. Intuitively, the number of births must equal the number of deaths over the long term.

Letting p_i represent the average fraction of time there are i customers in the system, the average rate at which the process crosses the vertical line from left to right is p_i, while the average rate at which it crosses it from right to left is $p_i + 1$. Hence,

$$p_i\lambda = p_{i+1}\mu$$

$$p_{i+1} = \frac{\lambda}{\mu} p_i = \rho p_i$$

This relationship holds for i = 0, 1, 2,.... so we can write

$$p_1 = \rho p_0$$
$$p_2 = \rho p_1 = \rho^2 p_0$$
$$p_3 = \rho p_2 = \rho^2 p_0$$
$$\cdots$$
$$p_n = \rho p_{n-1} = \rho^{n-1} p_0$$
$$\cdots$$

If we knew p_0 (i.e., the probability of an empty system) we would be able to compute all of the other long term probabilities. To compute this, we note that since the system must be in some state, the long term probabilities must sum to one. This implies the following.

$$1 = \sum_{n=0}^{\infty} p_n = \sum_{n=0}^{\infty} \rho^n p_0 = \frac{p_0}{1-\rho} \quad \ldots(2)$$

This implies that the probability of finding the system empty is

$$p_0 = 1 - \rho \quad \ldots(3)$$

and hence the long term probability of being in state n is given by

$$p_n = (1-\rho)\rho^n,\ n = 1, 2,... \quad \ldots(4)$$

This is known as the geometric distribution, whose mean is given by[4]

$$L^{M/M/1} = \sum_{n=0}^{\infty} n p_n = \sum_{n=0}^{\infty} n(1-\rho)\rho^n = \frac{\rho}{1-\rho} \quad \ldots(5)$$

So, $LM/M/1$ represents the average number of customers in the system (i.e., in service or waiting in queue). Again, this is only well-defined if <1.

Finally, we note that the expected number of customers in service is given by the probability that there are one or more customers in the system, which is equal to 1-p0 (i.e., one minus the probability that the system is empty.) Hence, the average number in queue can be computed as

$$L_q^{M/M/1} = L - (1 - p_0) = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho} \quad ...(6)$$

We can apply Little's Law (to compute the expected waiting time in the system and in the queue as follows:

$$W^{M/M/1} = \frac{L}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{\tau}{(1-\rho)} \quad ...(7)$$

$$W_q^{M/M/1} = \frac{L_q}{\lambda} = \frac{\rho^2}{\lambda(1-\rho)} = \left(\frac{\rho}{1-\rho}\right)\tau \quad ...(8)$$

Note: Birth-death process models do not consider twins (i.e., increases by more than one) or multi-fatality accidents (i.e., decreases by more than one).

Q. 6. What do you mean by the M/G/1 queuing model? Explain. **[UTU: 2012-13]**

Ans. The *M/M*/1 queuing model gives us some important insights. In particular, it shows that variability causes waiting and that waiting increases with utilization. But because it assumes that both inter arrival times and service times are Markov (exponentially distributed), there is no way to adjust the amount of variability in an M/M/1 system. To really understand how variability drives the performance of a queuing system, more general model is needed. The simplest option is the so-called M/G/1 queue, in which inter arrival times are still Markov but service times, are general.

To express and examine the M/G/1 queuing model, we need to characterize the variability of the service times. The most common measure of variability used in probability and statistics writings is the standard deviation, which we usually denote by σ (sigma). While this is a perfectly valid measure of variability, since it measures the "spread" of a probability distribution, it does not completely characterize variability of a random variable. To see why, suppose we are told that the standard deviation of a service time is five minutes. We cannot tell whether this constitutes a small or large amount of variability until we know the mean service time. For a system with a mean service time of two hours, a five minute standard deviation does not represent much variability. However, for a system with a two minute average service time, a five minute standard deviation represents a great deal of variability. To provide a more general measure of variability, we define the coefficient of variation (CV) of a random variable to be the standard deviation divided by the mean. So, the service time CV, denoted by cs, is given by

$$C_s = \frac{\sigma}{\tau}$$

Where σ represents the standard deviation of the service time and τ represents the mean, service time.

In the M/G/1 queue, however, cs can be anything from zero (which indicates deterministic service times with no variability at all) on up. This allows us to adjust variability and examine the impact on system behaviour. Deriving the expression for mean waiting time in the M/G/1 cannot be done using the birth-death approach used for the M/M/1 case. The reason is that since service times are no longer memory less, knowing the number of customers in the system is not sufficient to predict future behaviour. We must also know how long the current customer (if any) has been in service.

Q. 7. State analytical result for M/M/1/n model. [UTU: 2012-13]

Ans. In order to find analytical result or to know the performance measure, we consider an M/M/1/N queuing system with balking, and server vacations. The assumptions of the system model are as follows:

(a) Customers arrive at the system one by one according to a Poisson process with rate. On arrival a customer either decides to join the queue with probability bn or balk with probability 1-bn when n customers are ahead of him (n = 0, 1,....,N–1), where N is the maximum number of customers in the system, and

$$0\_bn+1\_bn < 1\ 1\_n\ \_\ N-1,$$
$$b0 = 1, \text{ and } bn = 0, n\_N$$

(b) After joining the queue each customer will wait a certain length of time T for service to begin. If it has not begun by then, he will get impatient and leave the queue without getting service. This time T is a random variable whose density function is given by

$$d(t) = \_e-\_t,\ t\ \_\ 0,\ \_ > 0$$

where _ is the rate of time T. Since the arrival and the departure of the impatient customers without service are independent, the average reneging rate of the customer can be given by (n-i). Hence, the function of customer's average reneging rate is given by

$$r(n) = (n-i)\_,\ i\ \_\ n\ \_\ N,\ i = 0, 1,$$
$$r(n) = 0,\ n > N.$$

(c) The customers are served on a first-come, first served (FCFS) discipline. Once service commences it always proceeds to completion. The service times are assumed to be distributed according to an exponential distribution with density function as sssfollows:

$$s(t) = \mu e-\mu t,\ t\ \_\ 0,\ - > 0$$

where _ is the service rate.

(d) Whenever the system is empty, the server goes on a sequence of vacations for a period of random time V. If the server returns from a vacation to find no customer waiting, he will begin another vacation immediately. It is assumed that V has an exponential distribution with the density function as follows:

$$v(t) = \_e-\_t,\ t\ \_\ 0,\ \_ > 0$$

where _ is the vacation rate of a server.

Q. 8. State M/M/C model of queuing theory. [UTU: 2012-13]

Ans. In the mathematical theory of random processes, the M/M/1 model is a multi-server queue model. It is a generalisation of the M/M/1 model.

Following Kendall's notation indicates a system where:

- Arrivals are a Poisson process
- Service time is exponentially distributed
- There are c servers
- The length of queue in which arriving users wait before being served is infinite
- The population of users (i.e. the pool of users), or requests, available to join the system is infinite

The waiting probability is also given by the Erlang C formula.

Average (expected) number of requests in the system:

$$\bar{N} = c\rho + \frac{\rho}{1-\rho}\pi_{c+}$$

Average (expected) length of the queue:

$$\bar{N}_Q = \frac{\rho}{1-\rho}\pi_{c+}$$

Average (expected) waiting time in the queue:

$$\bar{W} = \frac{\rho}{\lambda(1-\rho)}\rho_{c+}$$

Q. 3. Explain the M/G/1 Model in detail.

Ans. M/G/1/∞/∞ represents a single server that has unlimited queue capacity and infinite calling population, while the arrival is still Poisson process, meaning the statistical distribution of the inter-arrival times still follow the exponential distribution, the distribution of the service time does not. The distribution of the service time may follow any general statistical distribution, not just exponential. Relationships are still able to be derived for a (limited) number of performance measures if one knows the arrival rate and the mean and variance of the service rate. However the derivations are generally more complex and difficult.

A number of special cases of M/G/1 provide specific solutions that give broad insights into the best model to choose for specific queuing situations because they permit the comparison of those solutions to the performance of an M/M/1 model.

Q. 10. What do you mean by multiple- servers queue?

Ans. Multiple (identical) servers queue situations are frequently encountered in telecommunications or a customer service environment. When modelling these situations care is needed to ensure that it is a multiple servers queue, not a network of single server queues, because results may differ depending on how the queuing model behaves.

One observational insight provided by comparing queuing models is that a single queue with multiple servers performs better than each server having their own queue and that a single large pool of servers performs better than two or more smaller pools, even though there are the same total number of servers in the system.

One simple example to prove the above fact is as follows: Consider a system having 8 input lines, single queue and 8 servers. The output line has a capacity of 64 Kbit/s. Considering the arrival rate at each input as 2 packets/s. So, the total arrival rate is 16 packets/s. With an average of 2000 bits per packet, the service rate is 64 Kbit/s/2000b = 32 packets/s. Hence, the average response time of the system is $1/(\mu - \lambda) = 1/(32 - 16) = 0.0625$ sec. Now, consider a second system with 8 queues, one for each server. Each of the 8 output lines has a capacity of 8 Kbit/s. The calculation yields the response time as $1/(\mu - \lambda) = 1/(4 - 2) = 0.5$ sec. And the average waiting time in the queue in the first case is $\rho/(1-\rho)\mu = 0.03125$, while in the second case is 0.25.

Q. 11. Explain some other queuing models.

Ans. There are many other important queuing models which are useful in networking.

M/M/k for k > 1. Multiple (k) servers:

- Good model of a link which is made up of multiple channels, either physically or through multiplexing (e.g. a T1 Carrier is typically time division multiplexed with k = 24).
- Has worse performance at lower loads than M/M/1 with same total capacity.

M/M/k/k for k 1. One or more servers, no buffers (except one in each server).

- Important model in circuit switched networks.
- Models a trunk line with k circuits available.
- Any customer (a call) which doesn't get a circuit is blocked (gets a busy signal).
- Blocking probability is given by the Erlang B (or Erlang Loss) Formula

$$P_B = \frac{\rho^k / k!}{\sum_{i=0}^{k} \rho^i / i!} \quad \rho \geq 0$$

M/G/1. Arbitrary service (packet length) distribution.

- Can still compute the mean number in the system via the Pollaczek-Khinchine (P-K) Formula

$$E(n) = \left(\frac{\rho}{1-\rho}\right)\left[1 - \frac{\rho}{2}(1 - \mu^2\sigma^2)\right] \rho < 1$$

where σ^2 is the variance of the service time distribution.

- Again, variability (randomness) causes delay.
- Can apply Little's Formula to get the mean delay.

M/D/1. Deterministic service times (packet length).

- Special case of M/G/1 with $\sigma^2 = 0$

$$E(n) = \left(\frac{\rho}{1-\rho}\right)\left(1 - \frac{\rho}{2}\right) \rho < 1$$

- Under heavy load ($\rho \approx 1$), M/D/1 has half the delay of an M/M/1.
- This is one motivation for fixed-packet-length systems like ATM.

Can also model and analyze other queuing systems

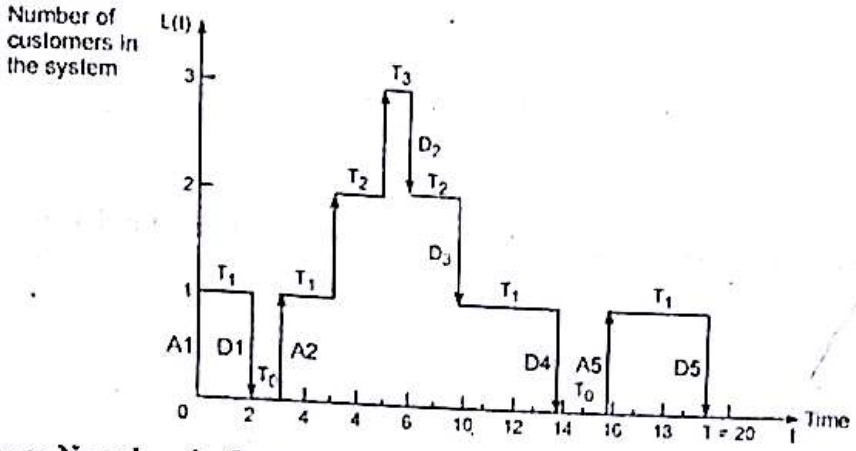
- With priority.
- With more general arrival processes.
- With "vacations."
- Many others.

Q.12. Explain any two long-run measures of performance of queuing models. [UTU: 2012

Ans. Long-run Measures of Performance of Queueing System

- Primary long-run measures of performance are
- L long-run time-average number of customers in system
- L_Q long-run time-average number of customers in queue
- W long-run average time spent in system per customer
- W_Q long-run average time spend in queue per customer
- ρ server utilization

Time-Average Number in System L



Time-Averge Number in System L

- Consider a queueing system over a period of time T
- Let T_i denote the total time during $[0, T]$ in which the system contained exactly i customers, the time-weighted-average number in thesystem is defined by:

$$\hat{L} = \frac{1}{T}\sum_{i=0}^{\infty} iT_i = \sum_{i=0}^{\infty} i\left(\frac{T_i}{T}\right)$$

- Consider the total area under the function is L(t), then,

$$\hat{L} = \frac{1}{T}\sum_{i=0}^{\infty} iT_i = \frac{1}{T}\int_0^T L(t)\,dt$$

- The long-run time-average number of customers in system, with probability 1:

$$\hat{L} = \frac{1}{T}\int_0^T L(t)\,dt \xrightarrow[T\to\infty]{} L$$

Average Time Spend in System Per Customer w

- The average time spend in system per customer, called the average system time, is:
- where $W_1, W_2, \ldots, W_N$ are the individual times that each of the N customers spend in the system during $[0, T]$.
- For stable systems: as $\hat{w} \to w$ as $N \to \infty$.
- If the system under consideation is the queue alone.

$$\hat{w}_Q = \frac{1}{N}\sum_{i=0}^{N} W_i^Q \xrightarrow[N\to\infty]{} w_Q$$

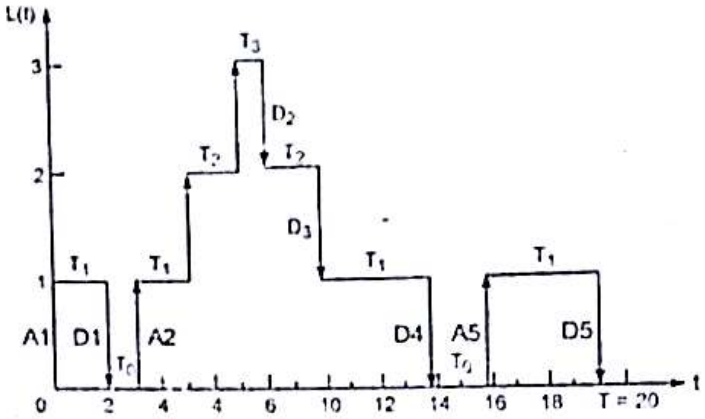
Averge Time Spend in System Per Customer w

$G/G/1/N/K$ example (cont.):

The averge system time is $(W_i = D_i - A_i)$

$$\hat{w} = \frac{W_1 + W_2 + \cdots + W_5}{5} = \frac{2 + (8-3) + (10-5) + (14-7) + (20-16)}{5} = 4.6 \text{ time units.}$$

The average queuing time is $\hat{w}_Q = \frac{0+0+3+3+0}{5} = 1.2$ time units.



Q.13. Develop a function translate ( ) to the simulation library that takes a random variable its range, and the range of the translated variable as input and returns value as a floating point number. [UTU: 2011-12]

Ans.

```
function translate(min, max)
{
   var ret;
   for (;;)
   {
      ret = min + ( Math.random() * (max-min)
* 1.1 );
      if ( ret <= max )
      {
         break;
      }
   }
   return ret;
   b=ret**-54;
   rand=ret-2*b;
   return rand;
}
```

Q.14. Let X be the random variable denoting the time to failure of a component. Suppose distribution function of X is $F(X)$. Use this distribution function to express the probability of following events. [UTU: 2011-

(a) $9 < X < 90$

(b) $X < 90$

(c) $X > 90$, given that $X > 9$

Ans. Let X be the random variable, the distribution function of X is $F(x)$.

(a) For $q < X < 90$

we know that

$$F(x) = P(X \le x)$$

$$= \sum_{t \le x} F(t) \quad \text{for } -\infty < x < \infty$$

where $F(t)$ is the value of the probability mass fuention of X at t.

$$F(x) = \begin{cases} 0 & : \ x < 9 \\ x & : \ 9 \le x < 90 \\ 1 & : \ 90 < x \end{cases}$$

(b) $F(x) = \begin{cases} 0 & : \ x > 90 \\ 1 & : \ x < 90 \end{cases}$

(c) $F(x) = \begin{cases} 0 & 9 < X < 90 \\ 1 & X > 90 \end{cases}$

Q.15. Show that if X has the k-stage Erlang distribution with parameter λ, then $Y = 2\lambda X$ has the chi-square distribution with $2k$ degrees of freedom. **[UTU: 2011-12]**

Ans. The probability density function of the Erlang distribution is

$$f(x; k, \lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} = \text{for } x, \lambda \ge 0$$

The parameter k is called the shape parameter and the parameter λ is called the rate parameter. An alternative, but equivalent, parametrization uses the scale parameter μ which is the reciprocal of the rate parameter (i.e., $\mu = 1/\lambda$):

$$f(x; k, \mu) = \frac{x^{k-1} e^{-x/\mu}}{\mu^k (k-1)!} = \text{for } x, \mu \ge 0$$

When the scale parameter equals 2, then distribution simplifies to the chi-squared distribution with $2k$ degrees of freedom. It can therefore be regarded as a generalized chi-squared distribution.

Because of the factorial function in the denominator, the Erlang distribution is only defined when the parameter k is a positive integer. In fact, this distribution is sometimes called the **Erlang-k distribution** (e.g., an Erlang-2 distribution is an Erlang distribution with $k = 2$). The Gamma distribution generalizes the Erlang by allowing k to be any real number, using the gamma function instead of the factorial function.

Cumulative distribution function (CDF)

The cumulative distribution function of the Erlang distribution is:

$$F(x; k, \lambda) = \frac{\gamma(k, \lambda x)}{(k-1)!}$$

where $\gamma(\ )$ is the lower incomplete gamma function. The CDF may also be expressed as

$$F(x; k, \lambda) = 1 - \sum_{n=0}^{k-1} \frac{1}{n!} e^{-\lambda x} (\lambda x)^n$$

Q.16. Simulate the following computer system in which jobs arrive at a CPU with an inter-arrival time that is normally distribution with a means of 4 seconds and variance 2 second. When a job is submitted it joins a queue. The CPU processes each jobs of FCFS basis. **[UTU: 2011-12]**

Ans. CPU Scheduling

A multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU.

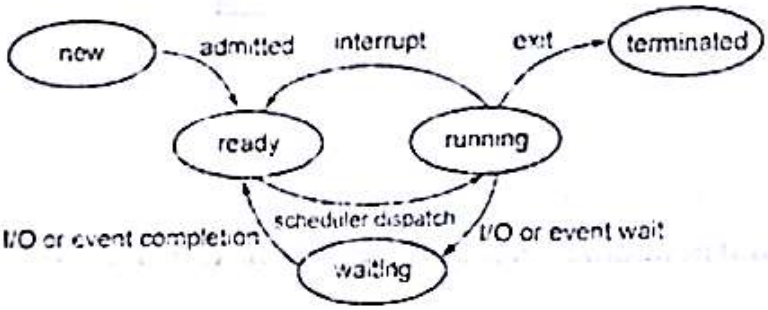


Fig *A diagram illustrating process state*

To make sure that scheduling strategy is good enough with the following criteria:

Utilization/Efficiency: keeps the CPU busy 100% of the time with useful work.

Throughput: maximizes the number of jobs processed per hour.

Turnaround time: from the time of submission to the time of completion and minimize the time batch users must wait for output.

Waiting time: Sum of times spent in ready queue.

Response Time: time from submission till the first response is produced and minimize response time for interactive users.

Fairness: make sure each process gets a fair share of the CPU.

FCFS - First-Come, First-Served

It is non-pre-emptive. Ready queue is a FIFO queue. Jobs arriving are placed at the end of queue. Dispatcher selects first job in queue and this job runs to completion of CPU burst. The advantages of FCFS is that it is simple and has low overhead. And has disadvantages of inappropriate for interactive systems and large fluctuations in average turnaround time are possible.

Simulation Algorithm

The job execution times are assumed to be drawn from a common distribution using exponential realistic for execution times. An exponentially-distributed random variable with parameter > 0 was used. The algorithm that generated the random number is shown below:

1. Public Function rndom() As Integer
2. Dim Lambda As Integer = 1

```
3. Dim seed As Single = 0
4. Dim Inter_Arrival = 4
5. Dim Variance = 2
4. Dim X As Integer = 0
5. Randomize()
6. seed = Rnd()
7. X = seed * Lambda
8. rndom = Int(1 - Exp(-Lambda * X) * 10)
9. Debug.Print(Exp(-Lambda * X))
10. Debug.Print(Exp(1))
11. End Function
```

Q.17. Develop a Program for the two sever queue simulation by assuming that in the event of both servers being free, the customer goes to the server that has been idle longer. Modify the code so that the customer always goes to server 1 if both servers are free. [UTU: 2011-12]

Ans. Consider a two-server system in which customers arrive in accordance with a nonhomogeneous Poisson process, and suppose that each arrival must first be served by server I and upon completion of service at 1 the customer goes over to server 2. Such a system is called a *tandem* or sequential queueing system. Upon arrival the customer will either enter service with server 1 if that server is free.

Suppose that we want to simulate the preceding model, keeping track of the amounts of time spent in the system by each customer, and the number of services performed by each server. Because there are multiple servers, it follows that customers will not necessarily depart in the order in which they arrive. Hence, to known which customer is departing the system upon a service completion we will have to keep track of which customers are in the system. So let us number the customers as they arrive, with the first arrival being customer number 1, the next being number 2, and so on. Because customers enter service in order of their arrival, it follows that knowing which customers are being served and how many are waiting in queue enables us to identify the waiting customers. Suppose that customers i and j are being served, where $i < j$, and that there are $n - 2 > 0$ others waiting in queue. Because all customers with numbers less than j would have entered service before j, whereas no customer whose number is higher than j could yet have completed service (because to do so they would have had to enter service before either i or j), it follows that customers $j + 1, \ldots, j + n - 2$ are waiting queue.

To analyze the system we will use the following variables:

Time Variable t

System State Variable (SS)

(n, i_1, i_2) if there are n customers in the system, i_1 is the server 1 and i_2 is with server 2. Note that $SS = (0)$ when the system is empty, and $SS = (1, j, 0)$ or $(1, 0, j)$ when the only customer is j and he is being served by server 1 or server_2, respectively.

Counter Variables

N_A: the number of arrivals by time t

C_j: the number of customers served by j, $j = 1, 2$, by time t

Output Variables

$A(n)$: the arrival time of customer n, $n \leq 1$

$D(n)$: the departure time of customer n, $n \geq 1$

Even list t_A, t_1, t_2

where t_A is the time of the next arrival, and t_i is the service completion time of the custom presently being served by server i, $i = 1, 2$. If there is no customer presently with server i, then set $t_i = \infty$, $i = 1, 2$. In the following, the event list will always consist of the three variables t_A, t_1

To begin the simulation, we initialize the variables and event list as follows:

Initialize

Set $t = N_A = C_1 = C_2 = 0$.

Set $SS = (0)$.

Generate T_0, and set $t_1 = T_0$, $t_2 = t_A = \infty$.

The update the system, we move along in time unit we encounter the next event. In t following cases, Y_i always refers to a random variable having distribution G_i, $i = 1, 2$.

Case 1: $SS = (n, i_1, i_2)$ and $t_A = (t_A, t_1, t_2)$

Reset: $t = t_A$.

Reset: $N_A = N_A + 1$.

Generate T_1 and reset $t_A = T_1$.

Collect the output data $A(N_A) = t$.

If $SS = (0)$:

Reset: $SS = (1, N_A, 0)$

Generate Y_1 and reset $t_1 = t + Y_1$.

If $SS = (1, j, 0)$

Reset: $SS = (2, j, N_A)$.

Generate Y_2 and rest $t_2 = t + Y_2$.

If $SS = (1, 0, j)$:

Reset $SS = (2, N_A, j)$.

Generate Y_1 and reset $t_1 = t + Y_1$.

If $n > 1$:

Reset $SS = (n + 1, i_1, i_2)$.

Case 2: $SS = (n, i_1, i_2)$ and $t_1 < t_A$, $t_1 \leq t_2$

Reset: $t = t_1$

Reset: $C_1 = C_1 + 1$.

Collect the output data $D(i_1) = t$.

If $n = 1$:

Reset: $SS = (0)$.

Reset: $t_1 = \infty$

If $n = 2$:

Reset: $SS = (1, 0, i_2)$.

Reset: $t_1 = \infty$.

If $n > 2$: Let $m = \max(i_1, i_2)$ and

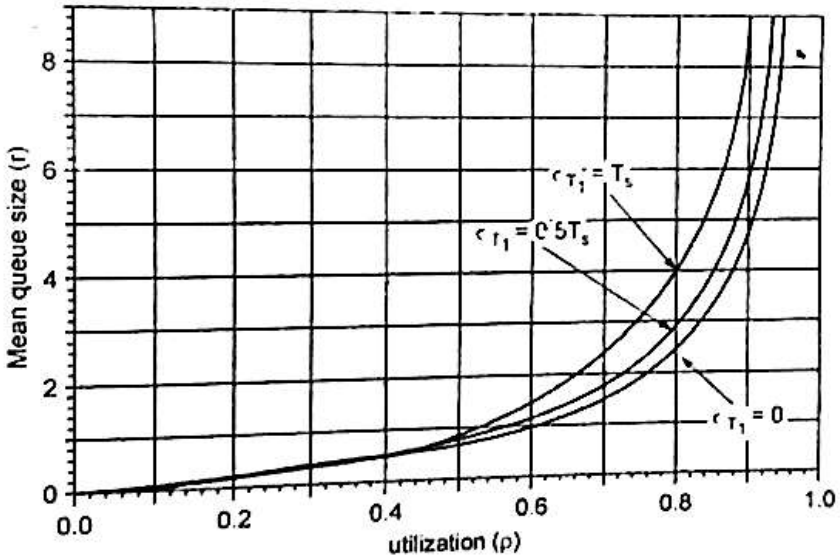
Reset: $SS = (n - 1, m + 1, i_2)$

Generate Y_1 and reset $t_1 = t + Y_1$

Case 3: $SS = (n, i_1, i_2)$ and $t_2 < t_A$, $t_2 < t_1$

Q.18. For a 1-server queue with Poisson arrival pattern and exponential service time, plot the following two quantities as function of the utilization factor P. (1) the average queue length and (2) the probability of queue length exceeding 5. Make sure to take enough points of higher values of P so those meaningful curves are obtained. **[UTU: 2011-12]**

Ans. The arrival rate is Poisson and the service time is general. Making use of a scaling factor, A, the equations for some of the key output variables are straightforward. Note that the key factor in the scaling parameter is the ratio of the standard deviation of service time to the mean. No other information about the service time is needed. Two special cases are of some interest. When the standard deviation is equal to the mean, the service time distribution is exponential (M/M/1). This is the simplest case and the easiest one for calculating results



❑❑❑

B.Tech.

TCS-506

(UTU) FIFTH SEMESTER EXAMINATION, 2013-14

MODELING AND SIMULATION

Time: 3 Hours

Total Marks: 100

Note: (1) Attempt all the questions.
(2) Each question carries equal marks.

1. Attempt any four parts of the following: (4×5=20)

Q.1. (a) List desirable features when selecting simulation software.

Ans.

1. When the process is well understood, can be defined by a flowchart and the operation times and rules can be described. This allows the modeler to study improvements without disturbing the actual system.
2. High volume or low volume with high capacity rate. Shipping terminals, warehouse picking and complex aircraft components are some excellent examples.
3. System complexity can be difficult or impossible to define on a spread sheet. Simulation allows looking at all system interaction and how they impact on each other.
4. Manufacturer, material handling, baggage handling, warehouse, supply chain, food processing, healthcare shipping ports and lean manufacturing implementation are just a few of the application model.

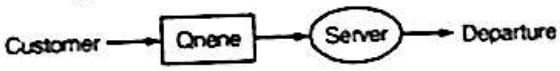
Simulation is a decision analysis and support tool. Simulation software allows evaluating, comparing and optimizing alternative design, plans and policies. Simulation should be used when the consequences of a proposed action, plan or design can not be directly and immediately observed or it is simply imprctical or prohibitively expensive to test the alternatives directly. For example, when implementing a stragetic plan for a company, the impacts are likely to take months or years to materialize. But use of simulation gives best results regarding time, cast and strategy. It is particularly valueable when there is significant uncertainty regarding the outcome or consequences of a particular alternative under consideration.

Probabilistic simulation allows dealing with this uncertainty in a quantifiable way. Most importantly, simulation, should be used when the system under consideration has complex interactions and requires the input from multiple disciplines. In this case, it is difficult for any one to easily understand the system. A simulation software model can act as the framework to integrate the various components in order to better understand the system and their interactions.

Q.1. (b) What are simulation requirements of a single server queue?

Ans. Simulation requirements of a single server queue:

In a single server queue, we consider that customer arrives randomly, waits for their turn to get service from service counter. The time which a customer takes for being served is called service time. When server is busy, customer joins the queue and queue works on first come first serve basis. As customer gets service, it leaves the queue.



First step is to gather information about it. Information about arrival pattern, service pattern, queue discipline, queue length, waiting time in queue and server utilization is necessary to model the queuing system.

In a single server queue there are only two type of events, arrival and service completion after which customers leave the queue.

We gather the information that whether customer arrives individually or in groups, what is the probability distribution of time between successive intervals and whether it is infinite or finite population. We need to know number of servers available, time distribution for service, separate queue for different servers or a single queue. Queue discipline gathered, it may be just come first served last come first served or random service.

Algorithm

1. Firstly set clock at time $t = 0$.
2. Initial system state; when server is idle sequence is empty.
3. Generate first arrival time.
4. Place it to future event list.
5. If server is idle, set server to busy.
6. Update server idle time statistic.
7. Place departure event in future event list.
8. If server busy, place customer in queue and update queue length statistic.
9. Select the next event; it may be next arrival or first served customer and then departure.
10. Advance simulation clock time to next event.
11. Execute the next event.
12. If queue is empty then set server to idle and update server idle time statistic.
13. Else remove the customer from queue, update queue waiting time statistic.
14. Generate the occurence time of the departure event and place it to list of future event list.
15. If not terminating, then simulation run go to step 9.
16. End program.

Q.1. (c) Give a layout for simulation of a telephone system.

Ans. Simulation of a telephone system: A key telephone system is a multiline telephone system used, in small offices and call centres. Key systems have individual line selection buttons for each connected phone line, however some common features are present for intercome dialing. It has independent keysets and electronic shared-control. Let us consider an example of call centre simulation modeling. It is used to optimize the performance of call centres. Different types of model are used to simulate call centres performance.

1. Interactive voice response model.
2. Voice broadcasting model.
3. Predictive dialer model.
4. Profit simulation model.

The IVR simulation model estimates the required resources for an automatic call center that answers inbound phone calls. The voice broadcasting simulator model accepts numerous input parameters to simulate the more complex process of broadcasting phone messages to individuals and answering machine. The predictive dialer simulator and autodialer model simulates outbound phone calls placed by a predictive dialing phone system with answered calls being connected to line agents. The call center profit simulator model uses the predictive dialer model in estimating profitability of a calling campaign using cost and profit estimate.

IVR simulation model: It is mainly used for computerized system that allows a person, to select an option from a voice menu and otherwise interface with a computer system. It starts interacting with customers with recorded message and options. The input data is either in form of voice input or touches keypad selections from menu. It is used in bank balance inquiry, call centers, customer satisfactory surveys etc.

Voice broadcasting simulation model: It is frequently used in industries to conduct surveys and to collect input data for a project. the processing of this model is to employ a computer to call one or more phone numbers from a list and to deliver phone messages. Simulation programming makes it sufficient to analyze the call and give output according to all that.

A call input may be a person, an answering machine, a busy signal etc. The computer determines what to do with each type of result. For a person or answering machine, the computer could leave a message. For a busy signals or no answers, the computer could schedule the call to be placed at a later time. It includes a number of parameters.

Q.1: (d) What are stochastic variables? Mention their properties.

Ans. Stochastic variable: A variable with uncertain quantity whose value depends on chance is called

stochastic variable or random variable. A random variable has a probability law a rule that assigns probabilities to the different values of the random variable. The probability law is called probability distribution of random variable.

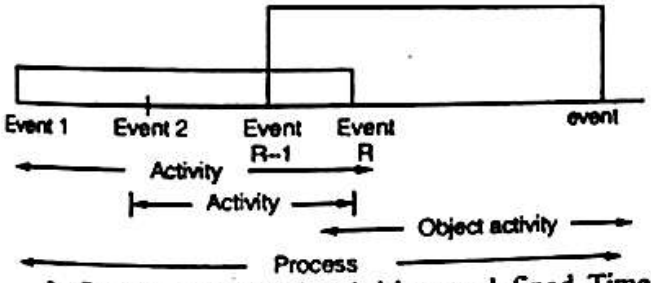
If x is a stochastic variable then probability distribution is denoted by $P(x)$. There are two types of stochastic variables:

1. Discrete and
2. Continuous random variable

Q.1. (e) Mention various components of a discrete event system simulation model.

Ans. Discrete event is related to occurrence at an instant of time. The purpose of a discrete event simulation is to study, a complex system by computing the time that would be associated with real events in a real life situation. For example, pushing an elevator button starting a motor, stopping of a motor and turning on a light are all discrete events because there is an instant of time at which event occurs.

The idea of discrete event simulation is to compute, as quickly as possible, the physical time that would occur in real time in physical system but without actually waiting for the delay between events to occur in real time.



In figure process and activities are defined. Time and state are very important aspects of discrete event simulation and different terms are associated with it. The system time, at which the value of at least one attribute of an object can be altered, is called an instant and duration between two successive instants is called interval. The act of counting al attributes values of an object is state, and state over an interval is an activity. Activity considered for a time intervals are not discrete events, for example moving train from point A to point B. To model this event we can consider two events, event of train leaving point A and arriving point B. If we can associate a time value with difference between the discrete event then we can model the duration activities as difference between end and starting point.

Q.1. (f) Write a note on deterministic vs stochastic simulation models.

Ans. Deterministic simulation model: Such type of simulation model assumes conditions of complete certainity and perfect knowledge. For example transportation, gas service, online exam, salary distribution etc.

Stochastic simulation model: These types of simulation models usually handle such situations in which consequences or pay off of maregerial action can not be predicted with certainity. For example:

1. Demand of items
2. Insurance company is willing to insure against risk of fire, accidents, sickness and so on because the pattern of evented have been compiled in the form of probability distribution.

2. Attempt any four parts of the following: **(4×5=20)**

Q.2. (a) Explain Little's theorem in queuing models with the help of an example.

Ans. (a) Little theorem: It is a statement of what was a 'folk theorem' in OR for many years.

$$\bar{N} = \lambda \bar{T}$$

When N is random variable for the number of jobs or customers in the system, λ is the arrival rate at which jobs arrive and T is the random variable for time a job spends in the system.

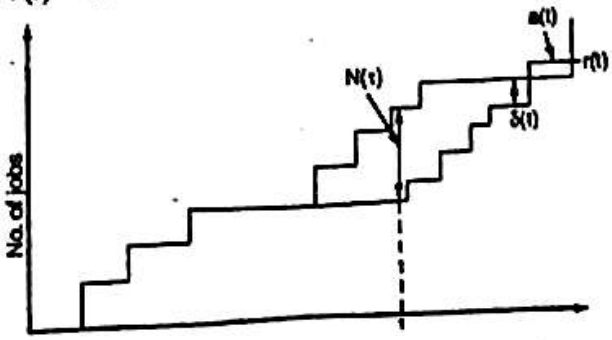
Let,

$a(t)$ = number of arrivals in the interval $[o, t]$

$\delta(t)$ = number of departures in interval $[o, t]$

$N(t)$ = number of jobs in the system at time $a(t) - \delta(t)$

$r(t)$ = accumulated customer λ second in $[o, t]$



$$\lambda(t) = \frac{a(t)}{t}$$

$\bar{N}_t$ = average number of jobs during $[o, t]$ time

$$= \frac{r(t)}{t}$$

$\bar{T}_t$ = average time of a job spent in system in $[o, t]$

$$= \frac{r(t)}{a(t)}$$

$$r(t) = \bar{T}_t\, a(t)$$

$$\bar{N}_t = \frac{\bar{T}_t\, a(t)}{t} \lambda(t)\, T_t$$

$$\lim_{t\to\infty} \lambda(t) = \lambda, \lim_{t\to\infty} \bar{T}_r = \bar{T}$$

$$\boxed{\bar{N} = \lambda \bar{T}}$$

Little theorem is applicable in estimating waiting time and planned inventory time.

Q.2. (*b*) Write technical notes on the following.

(*i*) Monte Carlo computation vs. Stochastic simulation.

Ans.

Monte Carlo computation:	Stochastic simulation
1. It is used to obtain solution of problems which are deterministic in nature.	1. It is used to obtain solution of problems which are stochastic in nature.
2. We can determine exact outcome at any time.	2. We can not deteremine exact outcome at any time.
3. Applies to static model.	3. Applies to dynamic model.
4. For example; Evaluating the constant π by making use of random numbers.	4. Example; The inventory problems.

(*ii*) Fixed time step vs. Event to event model.

Ans. Fixed time step model: It follows the characteristics such as:

1. In fixed time step model a 'timer' or 'clock' is simulated by the computer.

2. This clock is updated by a fixed time interval and system is examined to see if any event has taken place during this time interval τ.

Event to event model: It follows the characteristics such as;

1. It shifts from event to event.

2. The system state does not change in between. Only those points in time are kept track of when something of interest happens to the system.

3. This model is preferred because we do not waste any computer time in scanning those points in time when nothing takes place.

Q.2. (*c*) What are major simulation software in manufacturing applications? Also discuss modeling system randomness.

Ans. Major simulation software in manufacturing applications:

The following factors considered by the company while selecting simulation software for manufacturing system:

1. There should not be any mismatch with the basic manufacturing process. The major modes of manufacturing are:
   (*a*) True job-shop-make to order, little or no repetition.
   (*b*) Job- lot made to order with repetition
   (*c*) Continuous process made to stock.
2. There should not be any mismatch with the cost accounting system. The company should evaluate the type of control procedures that would best serve the needs of business.
3. Software should provide alternate path for manufacturing. This is because in reality the process routing may vary a great deal whereas, the software may assume a fixed path.
4. With selecting software, the company should incorporate flexibility to change the manufacturing environment. This will make

the software compatible with manufacturing processes.

5. The software should consider the scarcity of resources and its direct treatment to a greater extent in the analytical models. This is because in most manufacturing facilities or material handling functions, the influence of direct or indirect resources is often the major influence on manufacturing requirements and operations planning.
6. Unpredictable behaviour of the manufacturing system like varying cycle times, varying activity times, highly unpredictable material handling and manufacturing components should be simulated.
7. The powerful capability of simulation software is to be directly incorporated the effect of catastrophic or status disturbuing events into the system model. The events may be the real time failure of machines or products lines. When such events occur the analyst should dynamically change the operating procedures and parameters of the system when those events occur.
8. Inefficient and obsolete software to simulate the system should not be considered. Common examples of inefficient selections are: 1. GPSS 1V/V 2 SIM SCRIPT 3. Fortan
9. The software should assist organisation in managing their day to day operations. The software should be visual, interactive and use simulation technology to help users schedule their production orders.

Modeling system randomness: Some sources of randomness in simulated manufacturing system are:

1. Arrivals of orders, parts or raw material.
2. Processing, assembly or inspection times.
3. Machine times to failure.
4. Machine repair times.
5. Loading/unloading times.
6. Setup times.

Sources of randomness encountered in practice are merely normally distributed.

Q.2. (d) Mention various features needed in programming discrete event simulation models. Give a detailed comparison of simulation packages with programming languages.

Ans. Various features needed in programming discrete event simulation model:

1. Generation of random numbers to represent uncertainty.
2. Process transformers to permit other than uniform random varieties to be used.
3. List processing capability, so that objects can be created, manipulated and deleted.
4. Statistical analysis routines to provide the discriptive summary of model behaviour.
5. Report generation, to provide the presentation of potentially large reams of data in an effective way for decision making, and
6. A timing executive or time flow mechanism. Many software tools can and do needed these requirements.

Comparison of simulation packages with programming languages: One of the most important decision a modeler or analyst must make in performing a simulation study concern is the choice of software. If the selected software is not flexible enough or is too difficult to use, then the simulation project may produce erroneous results or may not even be completed. The following are some advantages of using a simulation package rather than a general-purpose proramming language.

1. Simulation package automatically provides most of the features needed to build a simulation model, resulting in a significant decrease in programming time and a reduction in overall project cost.
2. They provide a natural framework for simulation modeling. Their basic modeling constructs are more closely akin to simulation than are those in general purpose programming language like C.
3. Simulation models are generally easier to modify and maintain when written in a simulation package.
4. They provide better error detection because many potential types of errors are checked for automatically. Since, fewer modeling constructs need to be included in a model, the change of making an error will probably be smaller. (Conversely, errors in a new version of a simulation package itself may

be difficult for a user to find, and the software may be used incorrectly because documentation is some time lacking).

5. Most modelers already know a programming language but this is often not the case with a simulation package.
6. Programming language may allow greater programming flexibility than certain simulation package.
7. Software cost is generally lower, but total project cost may not be,

Although there are advantages to using both types of software, we believe in general that a modeler would be prudent to give serious consideration to using simulation package.

Q.2. (e) List and explain various measures of performance for queueing systems.

Ans. Various measures of performance for queuing system:

1. L_S represents expected length of customer in system an L_q represents expected length of customer in queue.

$$L_S = \sum_{n=0}^{\infty} n P_n, \qquad L_q = \sum_{n=s}^{\infty} (n-s) P_n$$

2. Traffic intensity or server utilization factor

$$l = \frac{\lambda}{\mu} = \text{Expected no. of customers in service.}$$

3. Expected number of customers in the system is equal to the expected number of customers in queue plus service time.
4. Expected waiting time of customers in the system is equal to the average waiting time in queue plus the expected service time.

$$L_S = L_q + \frac{\lambda}{\mu}, \quad W_S = W_q + \frac{1}{\mu}$$

5. Probability of only waiting for service longer than time t, is $P(T > t) = \frac{\lambda}{\mu} e^{-(\mu-\lambda)t}$
6. Probability of n customers in the system is

$$P_n = \left(1 - \frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^n$$

7. Probability that number of customers are increased by r, $P(n > r) = \left(\frac{\lambda}{\mu}\right)^{n+1}$
8. Relation between waiting time and length of customers in $L_S = \lambda W_S$ and $L_q = \lambda W_q$

Q.2. (f) Mention various advantages, disadvantages and pitfalls of simulation.

Ans. Advantages of simulations:

1. Simulation reduces the cost of experimentation on a system. A simulation model can evaluate proposed design, specifications and changes without commiting resources to build the actual system.
2. It predicts future outcome of a system.
3. It helps in understanding the cause and effect relationships in a system by dynamic system modeling.
4. It identifies bottlenecks, constraints and barriers in a system.
5. It provides repeatability to study a system by running the simulation model multiple times.
6. Simulation speeds up the run time of a model.
7. Designing and analyzing inventory control system.

Disadvantages

1. Model used to study large scale system tends to be very complex and writing computer programs to execute them can be a tedious task.
2. These models are usually expensive and time consuming to develop.
3. Level of model detail is inadequate.
4. Failer to collect good system data.
5. People don't have knowledge about simulation methodology.
6. Large number of variables makes simulation unwieldy and difficult.

Pit falls of simulation

1. Failure to have clearly defined objective at the begining of the study.
2. Inappropriate level of modeling details.

3. Failure to communicate with management on regular basis.
4. Treating it as if it is a complex exercise in computer pogramming.
5. Using commercial simulation software knowing its limitations.
6. Misuse of animation.
7. Failure to collect good system data.

3. Attempt any two parts of the following: **(2×10=20)**

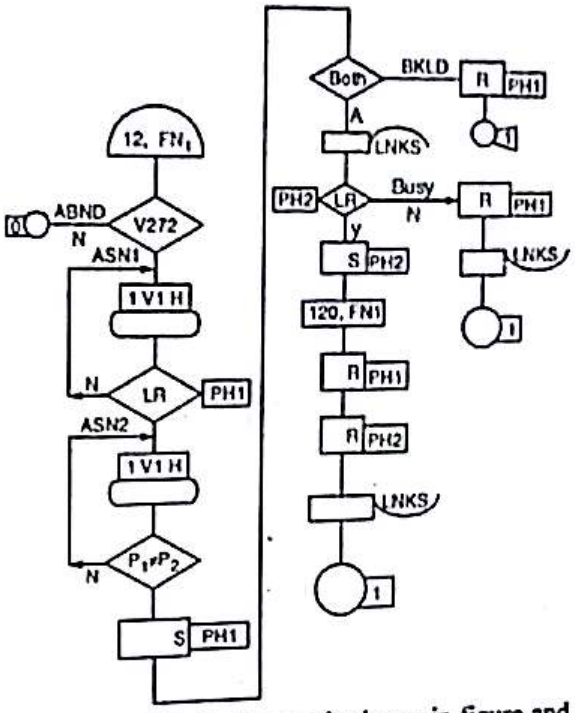
Q.3. (a) What are simulation languages? Explain any example using GPSS.

Ans. Simulation language: Simulation languages were general in nature and model development was done by writing code. This language provides a great deal of modeling flexibility, but were often difficult to use. Vendors of simulation language have attempted to make their software easier to use by employing a graphical model building approach. Example;

Simple telephone system with GPSS: To illustrate the use of logic switches, a GPSS model of telephone system is derived. The system is one in which a series of calls come from a number of telephone lines and the system is to correct the calls by using one of a limited number of links. Only one call can be made to any one line at a time and it is assumed that calls are lost if the called party is busy or no link is available. Each line is represented by a logic switch whose number is the line number.

The line is considered busy if the switch is set. Each call is represented by one transaction; the unit of time chosen is 1 second. It will be assumed that the distribution of arrivals is Poisson with a mean inter arrival time of 12 seconds. The length of the calls will also be assumed to have an exponential distribution.

As in the previous examination of this system, it will be assumed that each new call can come from any of the non-busy lines with equal probability and that its destination is equally likely to be any line other than itself.



A GPSS block diagram is shown in figure and coding are given below:

1.	Generate	12, FN1	Create calls
2.	Test G	V2, 2, ABND	Test if system is full
3.	ASN1 Assign	1, V1, PH	Pick origin
4.	Gate LR	PH1, ASN1	Test for busy
5.	ASN2 Assign	2, V1, PH	Pick destination
6.	Test NE	PH1, PH2, ASN1	Retry if test = origin
7.	Logic S	PH1	Make origin busy
8.	Transfer	Both, BLKD	Try for link
9.	GETL Enter	LNKS	Get link
10.	Gate LR	PH2, Busy	Test for busy
11.	Logic S	PH2	Make test busy
12.	Advance	120, FN 1	Talk
13.	Logic R	PH1	Origin Hang up
14.	Logic R	PH2	Destination Hang up
15.	Leave	LNKS	Free Link
16.	Term terminate	1	
17.	ABND terminate	1	ABANDON Call
18.	BLKD Logic R	PH1	Origin hangs up

19.	Terminate	1	Blocked calls
20.	Busy Leave	LNKS	Free link
21.	Logic R	P111	Origin hangs up
22.	Terminate	1	Busy calls
	LNKS Storage	10	Number of links
	1. Variable	XH1*RN1/1000+1	Pick a line
	2 Variable	XH1-2*SLNKS-2	Count no. of free line
	Initial	XH1, 50	Set no. of lines
	Start	10, NP	Initialize
	Reset reset		
	Start	1000	Main run

A generate block is used to create a series of transactions representing calls. The modifier at the block is the same function. No. 1 used in the supermarket example. The mean of GENERATE block is set to 12. Parameter 1 and 2 will be used to carry the origin and destination of the call.

Q.3. (*b*) What are Pseudo Random Numbers? Explain any one method of generating it?

Ans. Pseudo random numbers: It is not possible to generating truly random numbers in a simulation. Only natural processes generate truely random numbers. If random members are generated using a known mathematical formula then they can be replicated, so they are not truly random numbers. If random numbers are generated using a known mathematical formula then they can be replicated, so they are not truly random anymore but having appearance of randomness, *i.e.*, their statistical properties are very close to the one of the truly random numbers. So, these numbers are called 'Pseudo random numbers'. 'Pseudo', because generating numbers using a known method removes the potential for true randomness.

One of the first pseudo number generators was the midsquare method.

Midsquare method to generate pseudo random numbers: This method starts with a fixed initial number say of 4 digit integer, called the seed. This number is squared and the middle 4 digit of this square become the second number. The middle digit of this second number are then squared again to generate the third number and so on. Finally we get realization from the uniform (0, 1) distribution after placement of the decimal point (*i.e.*, after division by 10000). The choice of the seed is very important as the example will show.

If $x_1 = 2421$

then $x_1^2 = 05861241$

$x_3 = 1665$

$x_3^2 = 02772225$

$x_4 = 7722$ and so one

One may come across the following situations.

(*i*) The series may vanish because a random number obtained is 0000.

(*ii*) A random number reproduc itself e.g, $x_7 = 7600$, $x_8 = 7600$, $x_9 = 7600$.

(*iii*) A loop occurs e.g, $x_{11} = 6100$, $x_2 = 2100$, $x_3 = 4100$, $x_4 = 6100$ and the process continues in this circle.

Q.3. (*c*) What are major simulation software in manufacturing applications? Also discuss modeling system randomness.

Ans. Is same as given in answer of Q-2 (*c*).

4. Attempt any two parts of the following: (2×10=20)

Q.4. (*a*) Describe the importance of Discrete Probability functions.

Ans. Importance of discrete probability functions:

1. It is used in estimation of reliability of system.
2. It is applicable in radar detection.
3. It solved the problem concerned with arrival pattern of defective vehicles in a workshop.
4. Telephone calls.
5. Patients in a hospital.
6. Demand pattern for certain space parts.
7. Number of fragments from a shell hitting a target.
8. Emission of radioactive particles.
9. Number of printing mistake at each page of the book.
10. Number of air accidents in some time.
11. Number of deaths in a district by rare disease.
12. Number of cars passing through a street in some time.

All of the above are discrete random variable problem. So, the discrete probability, distribution (function) such as binomial, Poisson, multinomial, geometric and hypergeometric functions are used to solve all above problems. So, the discrete probability function is an important factor for the real life situations.

Q.4. (*b*) Describe GPSS in detail and also state its important features.

Ans. GPSS: The system to be simulated in GPSS is described as a block diagram in which the blocks

represent the activities and lines joining the blocks indicate the sequence in which the activities can be executed. Where there is a choice of activities, more than one line leavels a block and the conditions for the choice is stated at the block. The use of block diagram to describe system is of course very familiar.

However, the form taken by a block diagram description usually depends upon the person drawing the block diagram. To base a programming language on this descriptive method, each block must be given a precise meaning. The approach taken in GPBS is to define a set of 48 specific block types, each of which represents a characteristic action of the systems. The program user must draw a block diagram of the system using only these block types.

Each block type is given a name that is descriptive of the block action and is represented by a particular symbol. To assist the reader the block diagrams drawn will name the block types although this is not usually done by a programmer familiar with GPSS. Each block type has a number of data fields. As the blocks are described, the field will be referred to as field A, B, C and so on, reflecting the order in which they are specified.

Moving through the system being simulated are entities that depend upon the nature of the system.

For example, a communication system is concerned with the movement of message, a road transportation system with motor vehicle, a data proceeding system with records and soon.

In the simulation, these entities are called transactions. The sequence of events in real time is reflected in the movement of transaction from block to block in simulated time.

Transactions are created at one or 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 simulataneously. The transfer of a transaction from one block to another occurs instantaneously at a specific time or when some change of system conditions occurs.

A GPSS block diagram can consist of many blocks upto some limit prescribed by the program. An identification number called a location is given to each block and the movement of transaction is usually from one block to the block with the next highest location. The locations are assigned automatically by an assembly program within GPSS so that, when a problem is coded, the blocks are listed in sequential order.

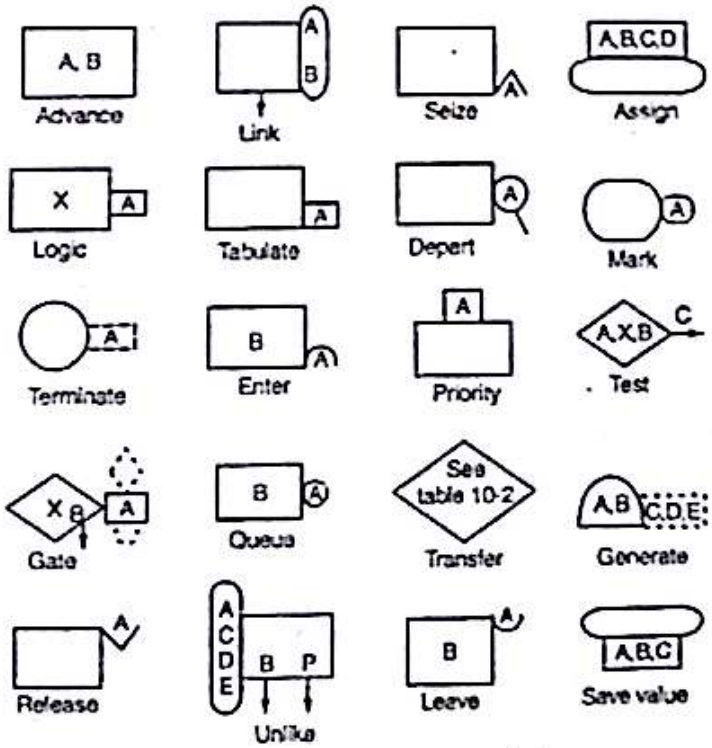


Fig. GPSS block diagram symbols

GPSS control statements

Location	Operation	A	B	C	D
	Clear end			$\left\{\begin{matrix}C\\D\\E\end{matrix}\right\}$	
Function no.	FUNCTION	Argument			
	Initial Job	Entity			
	Reset				
	simulate				
	start	Run count			
Storage no.	Storage	Capacity			
Table no.	Table	Argument		Laser limit	Interval No. of Intervals

Value

Q.4. (c) Describe the techniques used for queuing and also discuss queuing discipline.

Ans. The basic queueing process can be described as a process in which the customer arrives for service at the service counter, waits for his turn in the queue if the server is busy in the service of the other customer and is served when the server gets free. Finally the customer leaves the system as soon as he is served. There are some factors in which queueing system is described well:

1. Input (Arrival pattern): The input describes the way in which the customers arrive and join the system. Generally the customer arrives in a more or less random fashion which is not worth making the prediction. Thus, the arrival pattern can best be described in terms of probabilities and consequently the probability distribution for inter arrival times or the distribution of number of customers arriving in unit time must be defined. Generally the arrival pattern of customers are poisson or completely random.

2. Service mechanism: It is specific when it is known how many customers can be served at a time, what is statistical distribution of service times and when service is available. It is true in most situations that service time is a random variable with the same distribution for all arrivals, but case occurs where there are clearly two or more classes of customers each with a different service time distribution. It may be constant or random.

3. Customer behaviour: The customer generally behaves in four ways:

(*i*) **Balking:** A customer may leave the queue because the queue is too long and he has no time to wait.

(*ii*) **Reneging:** This occurs when a waiting customer leaves the queue due to impatience.

(*iii*) **Priorities:** Some customers are served before others regardless of their order of arrival. These customers have priorities over others.

(*iv*) **Jocking:** Customers may jockey from one waiting line to another. It may be seen that it occurs in supermarket.

4. Queue discipline: It is the rule determining the formation of the queue the manner of the customers behaviour while waiting. The simplest discipline is 'FCFS' such as at ration shop, at cinema ticket window. An extremely difficult queue discipline is service in random order. some notations are: PIFO, FCFS, LIFO, EILO, SIRO.

5. Attempt any twp parts of the following: **(2×10=20)**

Q.5. (*a*) Explain the following:

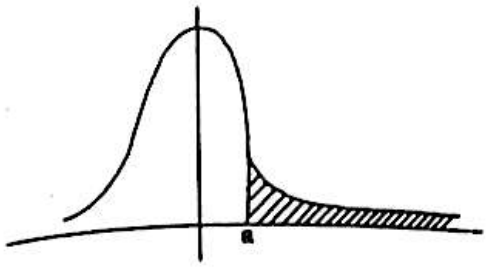
(*i*) Continuous probability functions

(*ii*) Discrete probability functions

Ans. (*i*) Continuous probability functions

If a random variable is a continuous variable its probability distribution is called a continuous probability distribution. A continuous probability function differs from a discrete probability function in several ways.

- The probability that a continuous random variable will assume a particular value is zero.
- As a result, a continuous probability distribution can not be expressed in tabular form.
- Instead an equation or formula is used to describe a continuous probability function.
- Since, the continuous random variable is defined over a continuous range of values (called the domain of the variable), the graph of the density function will also be continuous over that range.
- The area bounded by the curve of the density function and the x-axis is equal to 1, when computed over the domain of the variable.
- The probability that a random variable assumes a value between a and b is equal to the area under the density function bounded by a and b, for example,



(ii) Discrete probability distribution

If a random variable is a discrete variable, its probability distribution (function) is called a discrete probability function.

With a discrete probability distribution, each possible value of the discrete random variable can be associated with a non-zero probability. Thus a discrete probability function can always be presented in tabular form.

Example: Suppose you flip a coin two times then you can have four possible outcomes HH, HT, TH and TT. Let the random variable X represent the number of heads that result from this experiment. The random variable X can only take on the value 0, 1, 2 so it is a discrete random variable.

Number of heads	Probability
0	0.25
1	0.50
2	0.50

Above table represents a discrete probability distribution because it relates each value of a discrete random variable with its probability of occurrence.

Q.5. (b) Describe the characteristics of good random number generator.

Ans. Characteristics of good random number generator.

1. The generated number should satisfy the property of uniformity.
2. The generated number should satisfy the property of independence.
3. The random number should be replicable.
4. It should take a long time before numbers start to repeat.
5. The routine should fast.
6. The routing should not require a lot of storage.

Q.5. (c) Explain in detail about block types and block symbols used in GPSS.

Ans. Basic blocks of GPSS:

1. **Advance:** This block is concerned with representing the expenditure of time. The program computes an interval of time called an action time for each transaction as it enters as ADVANCE block.
2. **Generate:** This block controls the interval between, successive arrivals of transactions and normally creating transaction from zero time.
3. **Terminate:** Transactions are removed from the simulation at terminate block.
4. **Transfer:** The transfer block allows some location other than the next sequential location to be selected.
5. **Adopt:** Change assembly set.
6. **Alter:** Test and modify transaction in a group.
7. **Assembly:** Wait for and destroy related transaction.
8. **Assign:** Modify transaction parameter.
9. **Buffer:** Place transaction on the current events chain behind its priority peers.
10. **Close:** End the data stream.
11. **Count:** Place count of entities into a transaction parameter.
12. **Depart:** Decrement content of a queue entity.
13. **Displace:** Change the next sequential block of a transaction.
14. **Enter:** Occupy or wait for storage units in a storage entity.
15. **Examine:** Test group membership.
16. **Execute:** Perform action specified by different blocks.
17. **Funavail:** Change status of a facility entity to 'not available'.
18. **Favail:** Change status of a facility entity to 'available'.
19. **Gate:** Test entity and modify transaction flow.
20. **Gather:** Wait for related transaction.
21. **Generate:** Create transaction and place on future events chain.
22. **Index:** Modify transaction parameter.

23. **Integration:** Turn the integration of a user variable on or off.
24. **Join:** Place a member into a numeric group.
25. **Leave:** Release storage units of a storage facility.
26. **Link:** Move transaction to user chain entity.
27. **Logic:** Modify logic switch entity.
28. **Loop:** Decrement prameter, jump to different block if result is non-zero.
29. **Mark:** Place value of system clock into transaction parameter.
30. **Match:** Wait for related transaction to reach conjugate match block.
31. **Msave value:** Assign value to matrix entity element.
32. **Open:** Initialize a data stream.
33. **Plus:** Evaluate PLUS expression and save result in parameter.
34. **Preempt:** Displace facility owner.
35. **Priority:** Modify transaction priority.
36. **Queue:** Increment content of a queue entity.
37. **Read:** Bring the next line of data from a data stream.
38. **Release:** Free facility entity.
39. **Remove:** Take a member out of Numeric or transaction group.
40. **Return:** Free facility entity.
41. **Savail:** Change status of storage entity to 'available'.
42. **Save value:** Assign a value to save value entity.
43. **Scan:** Test transaction group, place value in parameter.
44. **Seek:** Change the line pointer in a data stream.
45. **Seize:** Assume ownership of or wait for facility entity.
46. **Select:** Place selected entity number into transaction parameter.
47. **Split:** Create related transaction.
48. **Sunavail:** Change status of storage entity to 'neet available'.
49. **Tabulate:** Update table entity.
50. **Terminate:** Destroy transaction, decrement termination count.
51. **Test:** Test arithmetic condition and modify transaction flow.
52. **Trace:** Set trace indicator of active transaction.
53. **Unlike:** Remove transaction from user chain entity.
54. **Untrace:** Turn off trace indicator in the active transaction.
55. **Write:** Send a value to a data stream.

□□□