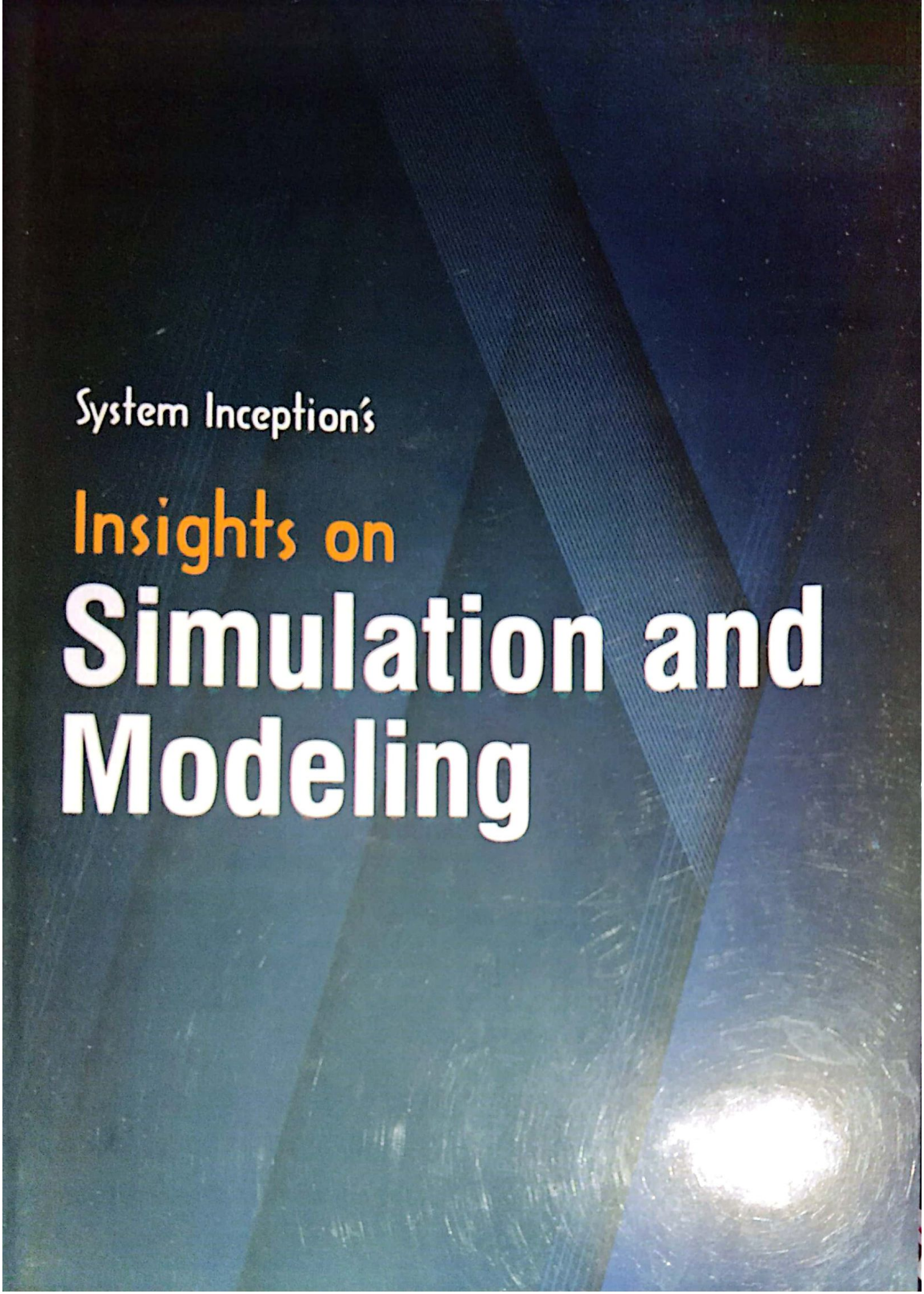


Insights on SIMULATION AND MODELING

By

Er. Susan Ghimire
Machine Learning Engineer,
Fusemachines Nepal Pvt. Ltd.

Er. Prem Chandra Roy
HoD, Department of Electronics and Computer Engineering,
ACEM, IoE, TU

Er. Tanisha Chaudhary
Data Engineer,
LIS Nepal Pvt. Ltd.

Er. Rabin Raj Gautam
DevOps Engineer

SYSTEM INCEPTION

Insights on Simulation and Modeling

Published by : **System Inception**

Authors : Er. Susan Ghimire
Er. Prem Chandra Roy
Er. Tanisha Chaudhary
Er. Rabin Raj Gautam

Edition : First, 2080 BS

Computer : Creation Graphics
Bagbazar, Kathmandu

Printed at : Solution Press Pvt. Ltd.
Kathmandu

CONTENTS

CHAPTER – 1

INTRODUCTION TO SIMULATION

CHAPTER – 2

PHYSICAL AND MATHEMATICAL MODELS

CHAPTER – 3

10

CONTINUOUS SYSTEM SIMULATION

CHAPTER – 4

QUEUING SYSTEM

CHAPTER – 5

MARKOV CHAIN

CHAPTER – 6

RANDOM NUMBER

CHAPTER – 7

VERIFICATION AND VALIDATION OF SIMULATION MODELS

CHAPTER – 8

ANALYSIS OF SIMULATION OUTPUT

CHAPTER – 9

SIMULATION SOFTWARE

CHAPTER – 10

SIMULATION OF COMPUTER SYSTEMS

CHAPTER 1

INTRODUCTION TO SIMULATION

CHAPTER – 1

INTRODUCTION TO SIMULATION

1.1 System, Model and Simulation

1.1.1 System

A *system* is defined as an organized set of interdependent objects working in a union to achieve a common goal. All systems have input, output, and feedback, and maintain a basic equilibrium level.

To understand a system, we should be familiar with the following terms:

- **Entity:** *Entity* denotes an object of interest in a system.
- **Attribute:** *Attribute* denotes a property of an entity. An entity can have many attributes.
- **Activity:** *Activity* is any process that causes changes in the system.
- **State:** *State* of the system is a description of all the entities, attributes, and activities as they exist at one point in time.

Examples of systems are computer systems, network systems, queuing systems, communication systems, etc.

1.1.2 Model

According to Geoffrey Gordon (System Simulation, 2nd ed.), a *model* is the body of information about a system gathered for the purpose of studying the system. In another word, a model is a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system. There is no unique model of a system. Different models of the same system can be generated by different analysts. Examples of different models are computer models of electronic systems, miniature models of vehicles, and mathematical models of supply and demand.

The table below is an example of a model of supermarket.

***Table:** Elements of a supermarket model*[1]

Entity	Attributes	Activities
Shopper	No of items	Arrive Get
Basket	Availability	Shop Queue Check-out
Counter	Number Occupancy	Leave Return

1.1.3 Simulation

A *simulation* is the imitation of the operation of a real-world process or system over time. Whether done by hand or on a computer, simulation involves the generation of an artificial history of a system and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system.

Some examples of simulation are weather simulation, fluid simulation, and simulation of queues in a supermarket.

Uses of simulation:

The uses of simulation are:

- To analyze the system before building, thus leading to a reduction in the number of design mistakes and optimizing the design.
- To analyze operational systems.
- To create a virtual environment for training and entertainment.
- To handle uncertainty.
- To analyze hazard and risks before real world operations.

When is simulation the appropriate tool?

Simulation can be appropriate tool for the following purposes:

1. Simulation is appropriate in the study of, and experimentation with, the internal interactions of a complex system or of a subsystem within a complex system.
2. Informational, organizational, and environmental changes can be simulated, and the effect of these alterations on the model's behavior can be observed.
3. The knowledge gained during designing a simulation model could be of great value in suggesting improvement in the system under investigation.
4. Changing simulation inputs and observing the resulting outputs can produce valuable insight into which variables are the most important and into how variables interact.
5. Simulation can be used as a pedagogical device to reinforce analytic solution methodologies.
6. Simulation can be used to experiment with new designs or policies before implementation, to prepare for what might happen.
7. Simulation can be used to verify analytic solutions.
8. Simulating different capabilities for a machine can help determine its requirements on it.
9. Simulation models designed for training make learning possible without the cost and disruption of on-the-job instruction.
10. Animation can show a system in simulated operation so that the plan can be visualized.
11. The modern system (factory, wafer fabrication plant, service organization, etc.) is so complex that its internal interactions can be treated only through simulation.

When is simulation not the appropriate tool?

Simulation is not advised in the following scenarios:

1. Simulation should not be used when the problem can be solved by common sense.

2. Simulation should not be used when the problem can be solved analytically.
3. Simulation should not be used when it is less costly to perform direct experiments.
4. Simulation should not be performed if the costs exceed the savings.
5. Simulation should not be used if the resources or time are not available.
6. Simulation is not advised if no data is available or if the ability to verify and validate the model is lacking.
7. Simulation should not be performed if the system behavior is too complex or cannot be defined.

1.2 Discrete and Continuous Systems

1.2.1 Discrete Systems

A *discrete system* is such in which the time evolution of the system is characterized by a sequence of discrete time steps. This means that the state of the system changes in discrete, distinct increments over time, rather than continuously. In other words, a discrete system can only take on certain values at specific points in time, rather than being able to take on any value within a continuous range. Examples of discrete systems include digital electronic circuits, cellular automata, and many mathematical models of physical systems.

1.2.2 Continuous Systems

A system is said to be *continuous* if the state variable(s) change continuously over time. *Continuous systems* are systems that are characterized by a set of continuous variables, rather than discrete variables. These systems can be described using a set of differential equations. Continuous systems are often used in modeling physical phenomena, such as the flow of fluids or the motion of objects. Because of the inherent complexity of these systems, they are typically analyzed using mathematical techniques

such as calculus and differential equations. Some examples include the oscillation of a spring-mass system, the flow of water through a pipe, and electric current through a circuit.

1.3 Model of a System

A *model* of a system is a simplified representation of that system, typically used to better understand its behavior or to make predictions about its future state. Models are often mathematical in nature and can take many different forms depending on the specific system being studied and the goals of the model. Some common types of models include conceptual models, which provide a high-level overview of the system; mathematical models, which use equations to describe the relationships between different variables in the system; and computational models, which use computer algorithms to simulate the behavior of the system.

1.3.1 Modeling

Modeling is the process of creating a model of a system. This typically involves identifying the key variables and relationships within the system and then using mathematical or computational techniques to represent those relationships in a simplified form. In other words, Modeling is the use of mathematical techniques, specialized hardware, and engineering judgment to create a representative stand for a real-world environment, device, system, or behavior. The goal of modeling is to better understand the system, to make predictions about its behavior, or to test different scenarios and see how the system might respond.

Modeling is an important tool in many fields, including engineering, physics, biology, economics, and computer science. It allows researchers and practitioners to analyze complex systems and make informed decisions based on their findings.

Modeling of a system can be categorized into two main tasks:

- **Establishing model structure:** Determines system boundaries, and identifies attributes, entities, and activities of the system.
- **Supplying the data:** Provide the values for the attributes and define the relationship involved in the activities.

1.4 Types of Models

Many different types of models can be used to represent a system, depending on the nature of the system and the goals of the model. Models can be classified into two distinct parts: that is Physical model and the Mathematical model. We can further classify the two main types of models as shown below.

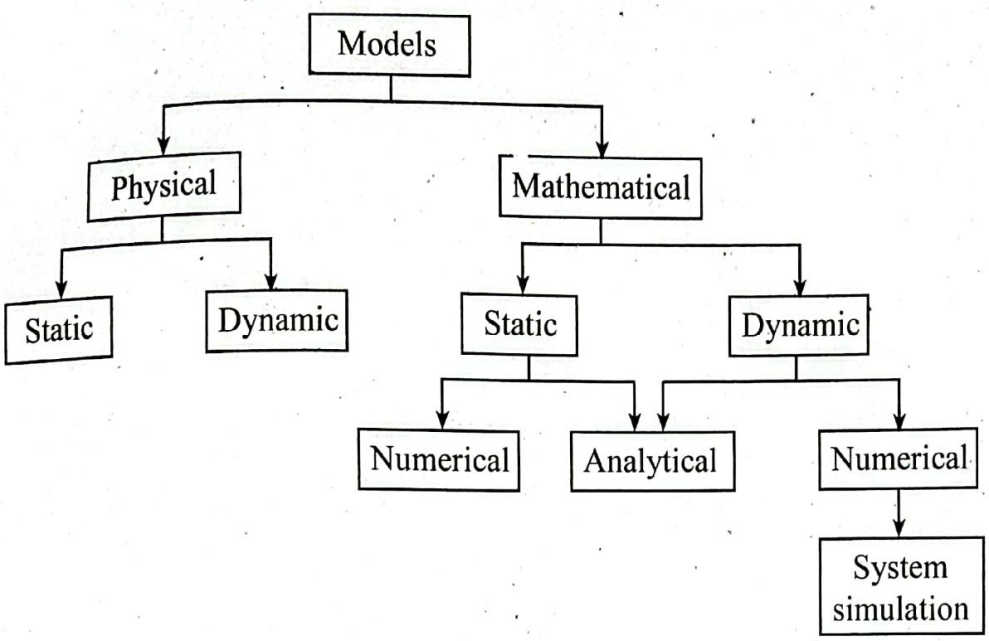


Fig.: Types of models

1.4.1 Physical Models

Physical models are models of a system that are physical objects, rather than mathematical or computational representations. These models can be used to represent a wide range of systems, from simple mechanical systems to complex biological or chemical processes. Physical models are based on some analogy between such systems as mechanical and electrical, or electrical and hydraulic. In a physical model of a system, the system attributes are represented by such measurements as a voltage or the position of a shaft. Physical model may be static or dynamic.

Physical models can take many different forms, depending on the specific system being modeled. Some examples of physical models include models of the solar system or other astronomical objects, mechanical or electrical devices that simulate the behavior of a system, etc.

Physical models can be useful because they allow people to see and tangibly interact with the system. This can help to better understand the system's behavior and make predictions about how it will respond to different stimuli. However, physical models can also be limited in their accuracy and complexity, compared to other types of models.

1.4.2 Mathematical Models

In *mathematical models*, the system attributes are represented by variables, and the activities are denoted by some mathematical functions depicting the relationship between variables. In mathematical models, symbolic notation and mathematical equations are used to represent a system. Mathematical model may be static or dynamic.

One advantage of mathematical models is that they can be very accurate, provided that the underlying assumptions are valid. They can also be used to make precise and quantitative predictions about the system's behavior. However, mathematical models can also be complex and difficult to understand, particularly for systems with many variables and interactions.

Examples of mathematical systems are:

- A model of the oscillations of a spring-mass system, which might involve equations for Hooke's law and the conservation of energy
- A model of the electrical current flowing through a circuit, which might involve equations for Ohm's law and Kirchhoff's laws

In each of these cases, the mathematical model is used to represent the relationships between the key variables in the system, and to make predictions about the system's behavior.

After formulating mathematical model, it is solved either by numerical method or analytical method. Analytical method is the use of deductive reasoning of mathematical theory while numerical method involves applying computational procedures.

1.5 Steps in Simulation Study

1.5.1 Steps in Simulation Study

Study of simulation comprises of steps as represented in the flowchart below:

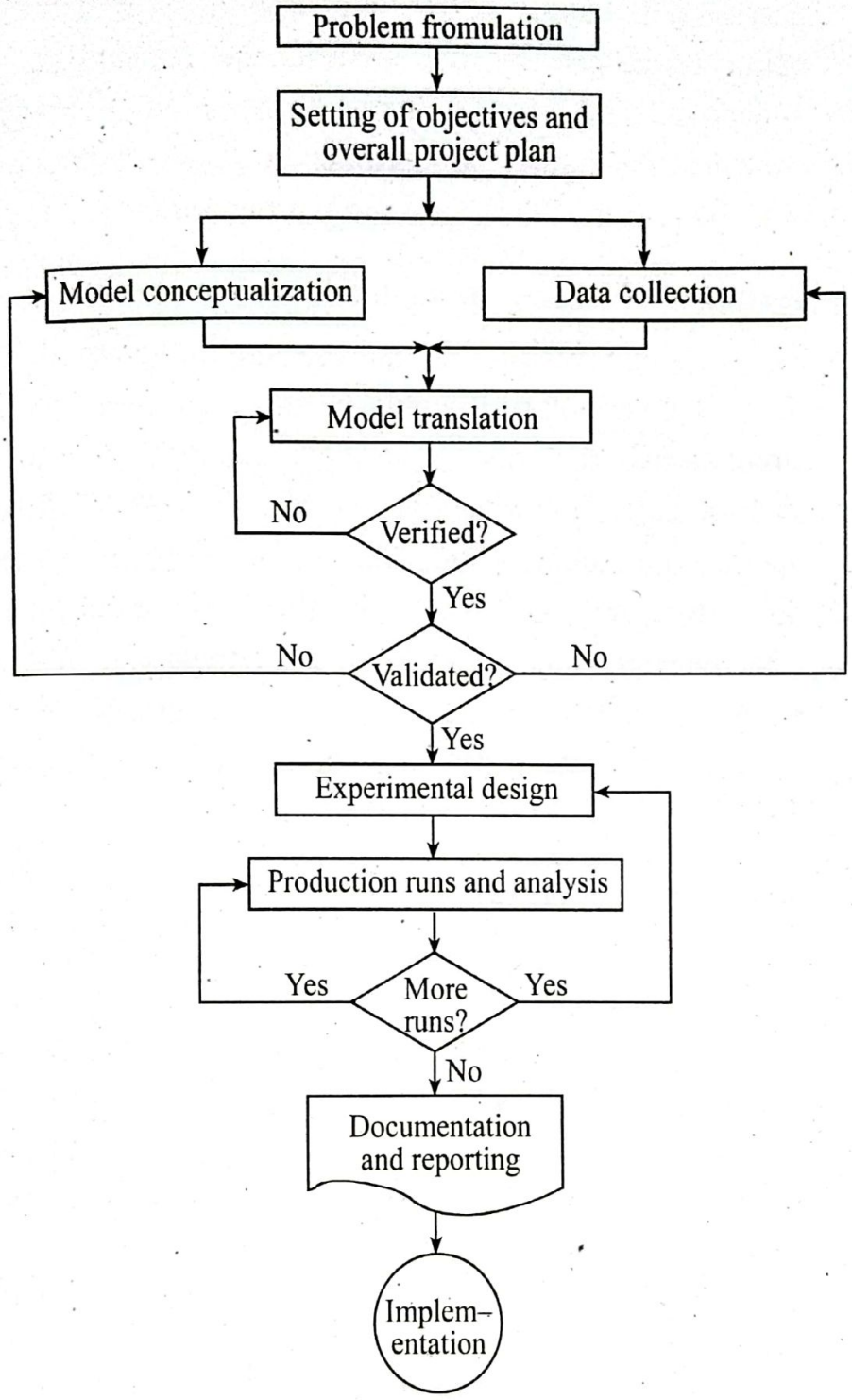


Fig.: Simulation study flowchart[3]

1. **Problem formulation:**

Every study should begin with a statement of the problem. If the statement is provided by policymakers or those that have the problem, the analyst must ensure that the problem being described is clearly understood. If a problem statement is being developed by the analyst, the policymaker must understand and agree with the formulation. Although not shown in the figure there are occasions where the problem must be reformulated as the study progresses.

2. **Setting of objectives and overall project plan:**

The objective indicates the questions to be answered. At this point, a determination should be made concerning whether simulation is the appropriate methodology for the problem as formulated and objectives as stated. The overall project plan should include a statement of the alternative systems to be considered and of a method for evaluating the effectiveness of these alternatives. It should also include the plans for the study in terms of the number of people involved, the cost of the study, the number of days required, etc.

3. **Model conceptualization:**

The construction of a model of a system is probably as much art as science. The art of modeling is enhanced by an ability to abstract the essential features of a problem and to select and modify basic assumption results. Thus, it is best to start with a simple model and build toward greater complexity. It is not necessary to have a one-to-one mapping between the model and the real system.

4. **Data collection:**

There is a constant interplay between the construction of the model and the collection of the needed input data. As the complexity of the model changes, the required data elements can also change. Also, since data collection takes such a large portion of the total time required to perform

simulation, it is necessary to begin it as early as possible, usually together with the early stages of model building. The objective of the study dictates, in a large way, the kind of data to be collected.

5. **Model translation:**

Most real-world systems result in models that require a great deal of information storage and computation, so the model must be entered into a computer-recognizable format. We use the word "program" even though it is possible to accomplish the desired result in many instances with little or no actual coding. The modeler must decide whether to program the model in simulation language, such as GPSS/H, or to use special-purpose simulation software.

6. **Verification:**

Verification is about checking if the computer program made for a simulation model is done correctly; Is the computer program performing properly? With complex models, it is difficult, if not impossible, to translate the model successfully in its entirety without a good deal of debugging; if the input parameters and logical structure of the model are correctly represented in the computer, verification has been completed. For the most part, common sense is used in completing this step.

7. **Validation:**

Validation usually is achieved through the calibration of the model, an iterative process of comparing the model against the actual system behavior and using the discrepancies between the two, and the insight gained, to improve the model. This process is repeated until model accuracy is acceptable.

8. **Experimental design:**

The alternatives that are to be simulated must be determined. Often, the decision concerning which alternatives to

simulate will be a function of runs that have been completed and analyzed. For each system design that is simulated, decisions need to be made concerning the length of the initialization period, the length of simulation runs, and the number of replications to be made of each run.

9. **Production runs and analysis:**

Production runs and their subsequent analysis are used to estimate measures of performance for the system designs that are being simulated.

10. **More runs:**

Given the analysis of runs that have been completed, the analyst determines whether additional runs are needed and what design those additional experiments should follow.

11. **Documentation and reporting:**

There are two types of documentation: program and progress. Documentation is necessary to understand how the model operates. Model users can change parameters if required to determine the relationship between input parameters and output measures of performance. The result of all the analyses should be reported clearly and concisely in a final report. This will enable model users to review the final formulation, the alternative systems that were addressed, the criterion by which the alternatives were compared, the results of the experiments, and the recommended solution to the problem.

12. **Implementation:**

The success of the implementation phase completely depends upon how well the previous eleven steps are performed. It is also contingent upon how thoroughly the analyst has involved the ultimate model user during the entire simulation process. If the model user has been thoroughly involved during the model-building process and if the model user understands the nature of the model and its

output, the likelihood of a vigorous implementation is enhanced.

1.5.2 Phases of Simulation Study

Different steps of the simulation study can be divided into 4 different phases:

1. **Phase 1 (Problem formulation):** This includes the problem formulation step.
2. **Phase 2 (Model building):** This includes model construction, data collection, programming, and validation of a model.
3. **Phase 3 (Running the model):** This includes experimental design simulation of runs& analysis of runs.
4. **Phase 4 (Implementation):** This includes documentation and implementation.

1.6 Model Development Life Cycle

In the simulation study, the model development life cycle can be divided into seven steps. It is a life cycle so these steps will be repeated continuously until the user demands are met. The flowchart below represents the model development life cycle.

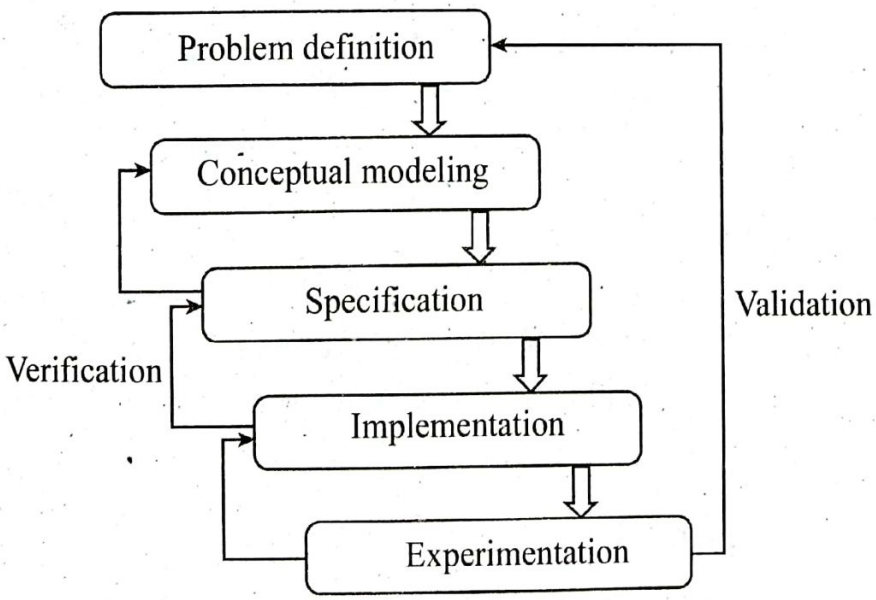


Fig.: Model development life cycle

1. **Problem definition:**

The first step in a simulation model development process is to define the problem that the model will be addressing, and to collect and prepare the data that will be used to build and evaluate the model.

2. **Conceptual modeling:**

Conceptual modeling is a process used to develop a high-level, abstract representation of a system or process. The goal of conceptual modeling is to provide a clear and concise representation of the system or process that can be used to facilitate communication and understanding among stakeholders and to serve as a basis for further analysis and design.

3. **Specification:**

The *specification* of a simulation model typically includes a description of the system being modeled, the input variables and their values, the equations or algorithms used to generate the model, and the output variables and metrics that will be used to evaluate the model's performance.

4. **Implementation:**

The developed model is implemented or we can also say the simulation begins in this step. The outputs are checked and verification is done to match the specification.

5. **Verification:**

It focuses on "Did I build the model right?" question. The computational model is tested to match the specification model. It largely involves software engineering activity (debugging).

6. **Experimentation:**

Different inputs are given to the model being simulated, variables are changed and the simulation is rerun. Different hypotheses are tested in the experimentation phase.

7. **Validation:**

It focuses on "Did I build the right model?" question. It is checked if the computational model matches the actual (or envisioned) system. We can only validate portions of the model. If the model can be validated 100% then there is no need to simulate the system in the first place.

1.7 Advantages and Disadvantages of Simulation

1.7.1 Advantages of Simulation

Jerry Banks et al. (Discrete-Event System Simulation, 5th ed., 2014) point out the following advantages of simulation:

1. New policies, operating procedures, decision rules, information flow or organizational procedures, and so on can be explored without disrupting the ongoing operation of the real system.
2. New hardware designs, physical layouts, transportation systems, and so on can be tested without physically building the system.
3. Hypotheses about how or why certain phenomena occur can be tested for feasibility.
4. Time can be compressed or expanded allowing for a speedup or slowdown of phenomena under investigation.
5. Insights can be obtained about the interaction of variables and the importance of variables to the performance of the system.
6. Bottleneck analysis can be performed to find out where work in process, information, materials, etc. is being delayed excessively.
7. Simulation aids to understand how the system operates.

1.7.2 Disadvantages of Simulation

According to Jerry Banks et al. (Discrete-Event System Simulation, 5th ed., 2014), following are the disadvantages of simulation:

1. Model building requires special training. Experts are rarely available for building the desired model. Moreover, training might take longer time than desired.
2. Simulation results might be difficult to interpret.
3. Simulation modeling and analysis might be time-consuming and even expensive, creating unnecessary financial problems.
4. Simulation is used in some cases when an analytical solution is possible or even more preferable.

1.8 Limitations of the Simulation Techniques

Simulation might not always be preferable and helpful for the study of a system. These are some cases where simulation fails to serve:

1. Simulation often requires a significant amount of computer time and is therefore expensive.
2. Simulation generates a way of evaluating solutions but does not generate solutions themselves.
3. Simulation is not precise. It does not yield an answer but merely provides a set of the system's responses to different operating conditions. In many cases, this lack of precision is difficult to measure.
4. It is a trial-and-error method that may produce different solutions in repeated runs.
5. The difficulty in finding the optimal values increases due to an increase in the number of parameters.

1.9 Areas of Application

Simulations are used in almost every field. These are some of the fields where it is extensively used:

1. **Manufacturing:**

   In manufacturing simulation can be used for design analysis and optimization of the production system, materials management, capacity planning, and performance evaluation of process quality.

2. **Business:**

   Simulation can be used for market analysis, prediction of consumer behavior and optimization of marketing strategy, comparative evaluation of marketing campaign.

3. **Military:**

   In military sector simulation can be helpful in testing of alternating combat strategies, air operations, and sea operations, simulating war exercises, inventory management, and virtual training.

4. **Healthcare:**

   In healthcare simulation is extensively used for planning health services, estimate expected patient density, facilities requirements, hospital staffing, and estimating the effectiveness of healthcare programs.

5. **Communication:**

   Network design, and optimization, evaluating the network reliability, and manpower planning are some application of simulation in communication sector.

6. **Economic:**

   Simulation can be used in economic sector for portfolio management, forecasting the impact of government policy and international market fluctuations on the economy, and forecasting market fluctuation.

7. **Transportation:**

Transportation sector uses simulation to design and testing of alternative transportation policies, transportation network- roads, railways, airways, etc; evaluation of timetables, and traffic planning.

8. **Environment:**

In environmental sector, simulation can be used for simulating solid waste management, performance evaluation of environmental programs, evaluation of pollution control system.

9. **Biological:**

Some of the biological applications of simulation are population genetics and the spread of epidemics, analyzing the problems with internal organs.

10. **Computer:**

Simulation is extensively used in computing for designing hardware configurations and operating system protocols, sharing networking, and client/ server architecture system.

REFERENCES

[1] Geoffrey Gordon. System Simulation. Second edition. Prentice Hall of India Private Limited, 2008, p.7.

[2] Jerry Banks et al. Discrete Event System Simulation. Fifth edition. Pearson Education Limited, 2014, p.4.

[3] Jerry Banks et al. Discrete Event System Simulation. Fifth edition. Pearson Education Limited, 2014, p.13.

Solution to Some Important Questions

1. Differentiate between deterministic system simulation and stochastic system simulation.

Ans:

Deterministic system simulation	Stochastic system simulation
A simulation consisting of no random elements with the known sets of inputs, resulting in unique output is known as deterministic system simulation.	A simulation containing random elements with one or more random input variables leading to random outputs is known as stochastic system simulation.
Output is deterministic for a given set of inputs.	Output is a random quantity (multiple runs are required to analyze the output).
Deterministic systems are often used in engineering and physical systems. Some examples are simulation of digital circuit, simulation of chemical based on a differential equation.	Stochastic systems are used in areas such as finance, economics, and epidemiology where uncertainty and randomness are important factors. Some examples are inter arrival time/service time of customers at a restaurant, amount of time to service a customer, queuing system.

2. Differentiate between the static simulation model and the dynamic simulation model.

Ans:

Static simulation model	Dynamic simulation model
Static simulation model represents the system that does not change with time but rather represents a system at a particular equilibrium state.	Dynamic simulation model represents the system changing with time representing a runtime model of a system.

Static simulation model	Dynamic simulation model
Time is not a significant variable.	Time is significant.
No use of the differential equation.	Use of differential equation.
More structural than behavioral.	Representation of behavior of static components of a system.
More rigid and cannot be changed over time. E.g.: Scale models	More flexible as could be changed with time. E.g.: differential equation models.

3. Differentiate between analytical solution and simulation solution.

Ans:

Analytical solution	Simulation solution
Analytical solution finds a closed-form expression (formula) that directly finds solution to the problem.	Simulation solution involves mathematical model and computational methods to approximate the solution of a problem through multiple iterations.
The solutions are precise and exact.	The solutions are approximate value (eg, mean, median, mode) of the multiple iteration. It is inherently subject to some degree of error.
Suitable for closed-form solution which can be mathematically modeled with simpler systems and known boundary conditions.	It is versatile and applicable to broader range of problems eg, dynamic and nonlinear systems.

Analytical solution	Simulation solution
It is quick and computationally less demanding.	It generally takes significant amount of time to process with high computational demand.
It is commonly used in simple systems, classical mechanics, theoretical models etc.	It is best suited for fluid dynamics, climate modeling, biological systems etc.

4. Differentiate between discrete simulation and continuous simulation.

Ans:

Discrete simulation	Continuous simulation
The system simulation in which state changes abruptly at discrete points in time is called discrete system simulation.	The system simulation in which the state of a system changes continuously with time is called continuous system simulation.
An event is associated with each transition state or timestamp.	The system is typically described by a set of differential equations.
Discrete simulation is often simpler and easier to implement.	Continuous simulation can be more complex and require more computational resources.
It may not provide as accurate a representation of the system being modeled as a continuous simulation.	It can provide a more accurate representation of the system being modeled.
The state of the system is updated after each event based on a set of rules or algorithms.	The state of the system is represented by a set of values for the variables, and the changes in the system over time are represented by changes in these values.

Exercise

1. Explain the steps of simulation study with flowchart. Differentiate between analytical solution and simulation solution. [2080 Shrawan]
2. Differentiate between simulation and modeling. Draw the flowchart representing phases in simulation study and explain the model building phase. [2079 Chaitra]
3. Define simulation and modeling. What are the advantages and disadvantages of simulation? [2079 Jestha]
4. Explain with various steps of simulation study. List the advantages of the simulation study. [2078 Chaitra, 2075 Bhadra]
5. What are the advantages of simulation? Explain briefly the steps in the simulation study. List the areas of application of simulation. [2077 Chaitra]
6. Define system, model, and simulation. Explain when is simulation considered an appropriate tool and when not? [2076 Bhadra]
7. What is simulation? Explain the steps of the simulation study with a flowchart. [2074 Bhadra, 2074 Magh]
8. Explain the different components of the system with examples. Write down the advantages and disadvantages of simulation. [2073 Bhadra]
9. Define system, model, and simulation. And clarify with a suitable example. List various advantages and disadvantages of simulation. [2073 Magh]
10. What are the major differences between stochastic simulation and deterministic simulation? [2071 Magh]

CHAPTER 2

PHYSICAL AND MATHEMATICAL MODELS

CHAPTER 2

PHYSICAL AND MATHEMATICAL MODELS

2.1 Static Physical Model

Before we proceed to static physical model, we should know about physical models in general. *Physical models* are smaller or larger copies of an object based on some analogy between mechanical and electrical, or electric and hydraulic systems. As said in the previous chapter, in the physical model of any system, the attributes are represented by such measurements as the voltage or position of the shaft. The physical model allows analysis of actual object by examining the characteristics of the represented model.

Examples of physical model are scale models, electrical stimulation of automobile wheels i.e., mechanical systems.

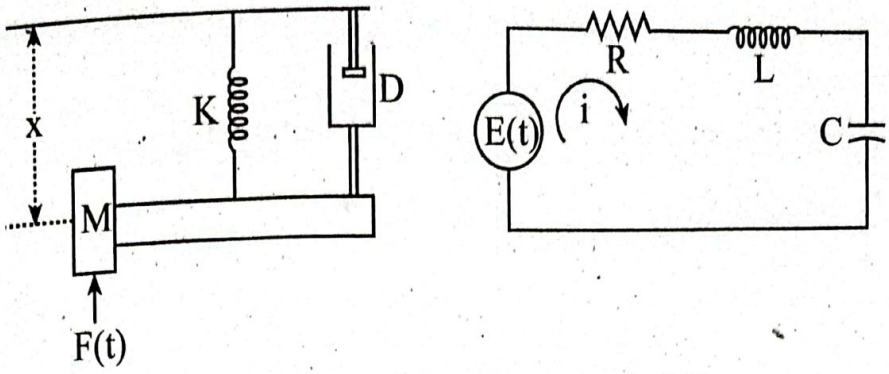
The model at the equilibrium state which shows the physical structure of the whole system attributes and their relationships are *static physical models*. Scale models are perfect examples of static physical models. A *scale model* is a physical representation of an object, which maintains the actual relationships between all system attributes. This model enables the demonstration of behavior or property of the original system without examining the system itself. For example, in ship building, making a scale model provides a simple way of determining the exact measurement of plates covering the hull, rather than having to produce drawings of many complicated shapes.

2.2 Dynamic Physical Model

The system models in which system activities change over time and rely upon an analogy between the system being studied and some other system of different nature, the analogy being dependent

upon underlying similarity in the forces governing the actual behavior of the system, are *dynamic physical models*.

In order to understand a dynamic physical model, consider a mechanical and an electrical system as shown in figure.



(a) *(b)*

Fig.: (a) Mechanical system (b) electrical system

Figure (a) might be a system representing the suspension of an automobile wheel when the automobile body is assumed to be immobile in the vertical direction. It consists of a mass M (that is subjected to time-varying force F(t)), a spring (whose force is proportional to its extension or contraction), and a shock absorber (that exerts a damping force proportional to the velocity of the mass. The motion of the system can be described by the differential equation

$$M\ddot{x} + D\dot{x} + Kx = K\,F(t) \quad \text{.....(i)}$$

where,

x is the distance moved by the wheel

M is the mass

K is the spring constant

D is the damping constant of the shock absorber

F(t) is the force applied

Figure (b) shows the electrical system having inductance L, resistance R and capacitance C, connected in series with time-varying voltage source E(t). The behavior of the system can also be described by the differential equation

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

where, q is the charge on the capacitance

Inspecting equations (i) and (ii), we find that the mechanical system (figure (a)) and the electrical system (figure (b)) are analogs of each other.

Mechanical system	Electrical system
Displacement, **x**	Charge, **q**
Velocity, $\dot{x}$	Current, **I** (= $\dot{q}$)
Force, **F(t)**	Voltage, **E(t)**
Mass, **M**	Inductance, **L**
Damping factor, **D**	Resistance, **R**
Spring constant, **K**	Inverse of capacitance, **1/C**

The performance of either system can be studied with the help of other. Since, practically, it is easier to modify the electrical system than to modify the mechanical system, it is always a wise decision to build the electrical system that will help to study the mechanical system. Any variation needed in a mechanical system can be checked by varying the electrical system.

For example, the behavior of a car wheel with changing damping factor (D) can be studied easily with analogous electrical system varying resistance (R) and observing the effect on voltage variation.

2.3 Static Mathematical Model

A *static model* gives the relationships between the system attributes when the system is in equilibrium. A *mathematical model* is a description of a system using various symbolic notations/mathematical language and equations. The system attributes are represented by variables and the activities are represented by some mathematical functions showing the

relationships between variables. A static mathematical model gives the relationship between the system attributes with mathematical relation/equations when the system is in equilibrium.

If the point of equilibrium is changed by changing any of the attribute values, the model can be used to derive the new values of all the attributes. But the model will not show how the attributes got new values. To understand this, consider an example in marketing a commodity.

In marketing a commodity, there is a balance between supply and demand for the commodity. When the price of the commodity is high, demand for the commodity will be low, and as the price decreases, demand for the commodity will increase. On the other hand, we can expect the supply of commodity to increase as the price of commodity increases. The price will settle to the point where the supply equals the demand. Taking price (P) as an independent variable, and demand (D), supply (S) as dependent variables, the graph will be as shown.

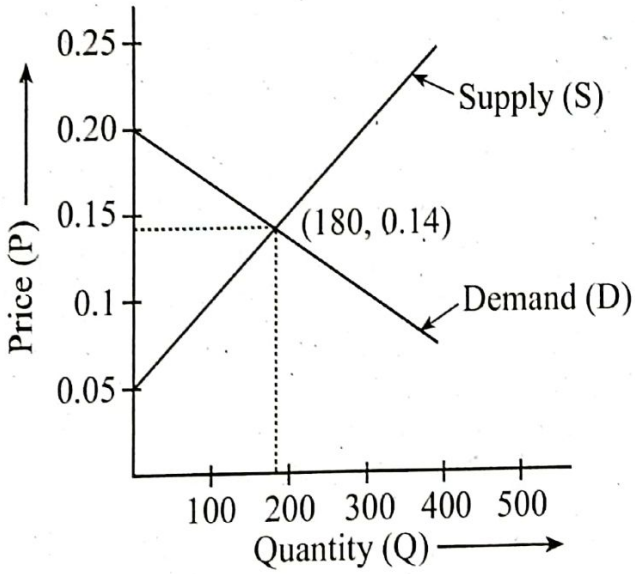


Fig.: Linear market model[1]

Assuming linear relationships between dependent and independent variables, mathematical model of the commodity market can be written as follows:

$$Q = a - bP$$

$$S = c + dP$$

$S = Q$

where a, b, d are positive constants, c is a negative constant.

At equilibrium, the market price can be calculated as

$$S = Q$$

$$\text{or, } c + dP = a - bP$$

$$\therefore P = \frac{a - c}{b + d}$$

The above graph shows the equilibrium price as 0.14 with the corresponding supply of 180.

Most often, the graphs of demand versus price and supply versus price are non-linear as shown.

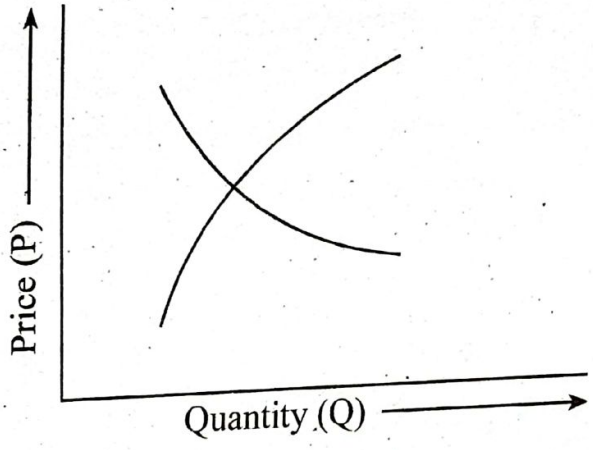


Fig.: Non-linear market model

On such occasions, some numerical method is required to solve the equations.

2.4 Dynamic Mathematical Model

A mathematical model that allows the changes of the system attributes to be derived as a function of time are *dynamic mathematical models*. Depending upon the complexity of the model, derivation might be analytic or numeric computation.

The equation,

$$M\ddot{x} + D\dot{x} + Kx = K\,F(t)$$

that describes the behavior of a car wheel is an example of a dynamic mathematical model.

Above equation can be written as

$$\ddot{x} + 2\zeta\omega\dot{x} + \omega^2 x = \omega^2 F(t) \qquad \text{.....(i)}$$

Here, ζ is the damping ratio, and $2\zeta\omega = D/M$, $\omega^2 = K/M$.

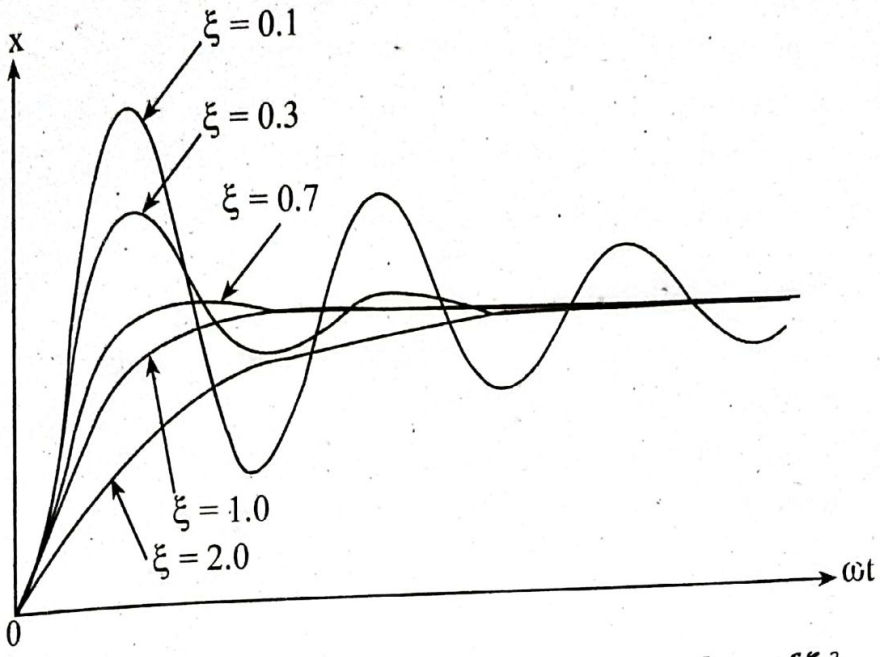


Fig.: Solutions of equation (i) for different values of ζ.[2]

The above plot shows how the position of the car wheel varies in response to steady-state force applied at time t = 0, that is, when a load is suddenly placed on the automobile. As revealed in the graph, motion is oscillatory when $\zeta < 1$ and the frequency of oscillation is given by:

$$\omega = 2\pi f$$

The relationship stated above between ζ, ω, M, K, and D provides insights on how to select the spring and shock absorber to get a certain type of motion. For example, in order to achieve motion without oscillation, we must have

$$\zeta \geq 1$$

$$\text{or, } \frac{D}{2\omega M} \geq 1$$

or, $D^2 \geq 4\omega^2 M^2$

or, $D^2 \geq 4MK$

This condition should be satisfied for motion without oscillation.

2.5 Principles used in Modeling

There is no specific rule to build mathematical models. However, a number of principles can guide us in building a model. This principle describes different viewpoints from which we can decide what information to put in the model. But these principles do not describe distinct steps carried out in building a model.

1. **Block building:**

   The description of the system should be organized into modular components or blocks, where each block represents a distinct part of the system. This allows for a clearer understanding of the system's structure and facilitates the analysis and modeling of individual components.

2. **Relevance:**

   Only those aspects of the system should be included in the models that are relevant to the study objectives. Irrelevant information increases the complexity of the model and will demand more effort to solve the model. For example, if a factory system study aims to compare the effects of different operating rules on efficiency, it is not relevant to consider the effects of hiring of employees as an activity.

3. **Accuracy:**

   Accuracy of information gathered for a model is another principle that should be taken into account. For example, in the aircraft system, an engineer responsible for estimating the fuel consumption may not require detailed description of the airframe. But an engineer responsible for comfort of the passengers needs to consider the vibrations of the aircraft, and thus requires detailed description of the airframe.

4. **Aggregation:**

Aggregation is another factor that has to be considered while building a model. Aggregation is the extent to which the number of individual entities can be grouped together into larger entities. This involves determining the appropriate level of detail at which the model operates and whether it is more effective to represent entities or activities individually or in aggregated form. For example, in a factory, a general manager may not want more description. However, a product control manager will have to consider the shops of all the departments as individual entities.

REFERENCES

[1] Geoffrey Gordon. System Simulation. Second edition. Prentice Hall of India Private Limited, 2008, p.14.

[2] Geoffrey Gordon. System Simulation. Second edition. Prentice Hall of India Private Limited, 2008, p.16.

Exercise

1. Explain the static mathematical model with a practical example. [2080 Shrawan, 2077 Chaitra]
2. Differentiate between static mathematical model and dynamic mathematical model with necessary mathematical equation and a practical example of each model. [2079 Chaitra]
3. Compare static and dynamic mathematical models with examples. [2079 Jestha, 2075 Bhadra, 2074 Bhadra]
4. Explain the dynamic physical model with practical examples. [2078 Chaitra, 2074 Magh, 2072 Magh]

CHAPTER 3

CONTINUOUS SYSTEM SIMULATION

CHAPTER - 3

CONTINUOUS SYSTEM SIMULATION

3.1 Continuous System Simulation

Continuous system simulation is a process of using computer software to model and analyze the behavior of a system over time. These models typically consist of sets of equations, such as differential equations and algebraic equations, that describe the behavior of the system over time. The simulation process allows us to study and analyze the system's dynamics, predict its future behavior, and understand how different inputs or parameters affect its output. It can be used to study a wide range of systems, including mechanical, electrical, chemical, and biological systems. It can be used to optimize the design of a system, predict its performance under different operating conditions, or identify potential problems or failures.

There are many different software tools available for continuous system simulation, including commercial packages such as Simulink, MATLAB, and Ansys, as well as open-source options such as Scilab and OpenModelica.

3.2 Differential Equations

A *differential equation* is a mathematical equation that relates some function with its derivatives, where functions usually represent physical quantities, the derivatives represent the rate of change and the equation represents the relation between two. In scientific and engineering studies, differential equations have a very important role. Differential equations are used to describe the behavior of systems that change over time, such as physical systems in engineering, biology, and physics. Differential equations may be *linear* or *nonlinear*.

The wheel suspension system of an automobile is given by:

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

This is an example of a linear differential equation with constant coefficients where t is an independent variable; x is a dependent variable; M, D, K are constants; and F(t) is an input.

Differential equations may also be *ordinary differential equations* or *partial differential equations*. If a differential equation involves a single independent variable, usually time, and one or more dependent variables, it is known as *ordinary differential equation*. If a differential equation contains more than one independent variables, it is known as *partial differential equation*. Partial differential equation can involve the derivatives of the same dependent variable with respect to each independent variables. The equation that describes the flow of heat in a three-dimensional body is an example of the partial differential equation. In this case, there are four independent variables (x, y, z, and t) and one dependent variable (temperature).

3.2.1 Significance of Differential Equations

The significance of differential equations can be listed as follows:

1. Differential equations allow us to mathematically describe the behavior of systems and predict how they will respond to different inputs and conditions.
2. Differential equations can capture the inherent complexity and nonlinearity of many real-world systems. This means they can provide a more accurate and detailed description of a system's behavior than other mathematical models, such as linear or algebraic equations.
3. Most physical and chemical process involves the rate of change which requires differential equation for their mathematical description.
4. Differential equations are extensively used to describe analytical solutions of a system.
5. Differential equations are extensively used in simulating the behavior of a real operational system using electrical circuits.

In addition to their use in modeling physical systems, differential equations are important in many other fields, including economics, biology, and engineering. They are used to study a wide range of phenomena, including the spread of diseases, the behavior of financial markets, and the dynamics of mechanical systems.

3.2.2 Importance of Differential Equations in Simulation Study

Differential equations play a crucial role in simulation studies because they provide a mathematical model for how physical systems change over time. They describe how a system evolves from its initial state to its final state, based on the laws of physics, biology, or other scientific disciplines. In many cases, these laws are formulated as differential equations that capture the rate of change of the system's variables, such as position, velocity, and temperature, as a function of time and other inputs.

The use of differential equations in simulation studies provides several important benefits. These are:

1. **Accurate modeling:**

   Differential equations provide a precise and accurate mathematical representation of a system, allowing for detailed and sophisticated simulations of complex processes.

2. **Flexibility:**

   Differential equations can be adapted to represent a wide range of physical and biological systems, making them versatile tools for simulation studies.

3. **Predictive power:**

   By solving differential equations, it is possible to make predictions about the behavior of a system, which can be used to optimize design, control processes, and make decisions.

4. **Efficient computation:**

   There are many numerical methods available for solving differential equations, and advances in computer technology

have made it possible to solve large and complex differential equation models in a reasonable amount of time.

In conclusion, the use of differential equations in simulation studies provides an essential tool for modeling and understanding complete systems and has numerous applications in fields ranging from engineering, physics, and biology to finance and economics.

3.3 Continuous System Model

A *continuous system* is one in which the predominant activities of the system cause smooth changes in the attributes of system entities. When such a system is modeled mathematically, the variables of models representing the attributes are controlled by continuous functions.[1]

Continuous system models are mathematical models that describe the behavior of systems that change continuously over time. These models are typically represented by differential equations, which describe how the variables in the system evolve.

Continuous system models are used to study a wide range of systems, including mechanical, electrical, chemical, and biological systems. They can be used to optimize the design of a system, predict its performance under different operating conditions, or identify potential problems or failures.

There are many different types of continuous system models, including linear and nonlinear models, deterministic and stochastic models, and static and dynamic models. The choice of model depends on the characteristics of the system being studied and the goals of the analysis.

3.4 Analog Computers

An *analog computer* uses continuous physical quantities, such as voltages or rotations, to represent and process data. It is used to process analog data. Before the advent of a digital computer, continuous system simulation was performed with an analog computer. Analog computer does not use discrete values but rather

continuous values. Examples of analog computers are slide rule, nomogram, oscilloscope, television, analog sound processor, etc.

The most widely used form of an analog computer is the *electronic analog computer* which utilized high-gain dc amplifiers called *operational amplifiers*. In such computer, voltages in the computer are taken as mathematical variables, and the operational amplifiers with appropriate circuits, can be made to add several input voltages (summer circuit), integrate a given voltage (integrator), or reverse the sign of input voltage (sign inverter).

3.4.1 Advantages and Disadvantages of Analog Computers

Some of the advantages of analog computers are:

1. One of the main advantages of analog computers is their ability to perform complex calculations very quickly.
2. They can directly work on physical quantities like voltage, force, displacement without any conversion.
3. Analog computers are also relatively simple to build and operate, compared to digital computers.
4. Analog computer does not suffer from quantization noise.

The disadvantages of analog computers are:

1. It is difficult to carry the accuracy of measuring a voltage beyond a certain point.
2. The assumption that there should be zero output voltage for zero input is difficult to achieve.
3. Operational amplifiers have a limited dynamic range of outputs, so a scale factor must be introduced to keep within range. Hence, it is difficult to maintain accuracy better than 0.1% in an electronic analog computer.
4. The computer is dedicated to solving one problem only.

Although analog computers were once widely used, they have largely been replaced by digital computers, which offer many advantages, including higher accuracy, greater flexibility, and easier programmability. However, some analog computers are still

used today for specialized applications, such as controlling industrial processes or simulating complex systems.

3.5 Analog Methods

3.5.1 Analog Computer Model for the Automobile Suspension Problem

Consider the differential equation:

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

$$\text{or, } M\ddot{x} = KF(t) - D\dot{x} - Kx$$

where, M is mass, x is distance moved, D is damping factor, K is stiffness of spring, and F(t) is force applied.

There are three variables in the equation namely F(t), $-\dot{x}$, and $-x$. Scaling these variables first and adding them with the help of a summer circuit will produce a voltage $M\ddot{x}$. Integrating the voltage $M\ddot{x}$ with a scale factor of 1/M will produce $\dot{x}$. When the voltage $\dot{x}$ is passed through an inverter, it will output $-\dot{x}$. Integrating $-\dot{x}$ will give $-x$. Finally, passing $-x$ through an inverter (sign changer) will give $+x$. Thus, a block diagram to solve above differential equation can be shown as follows.

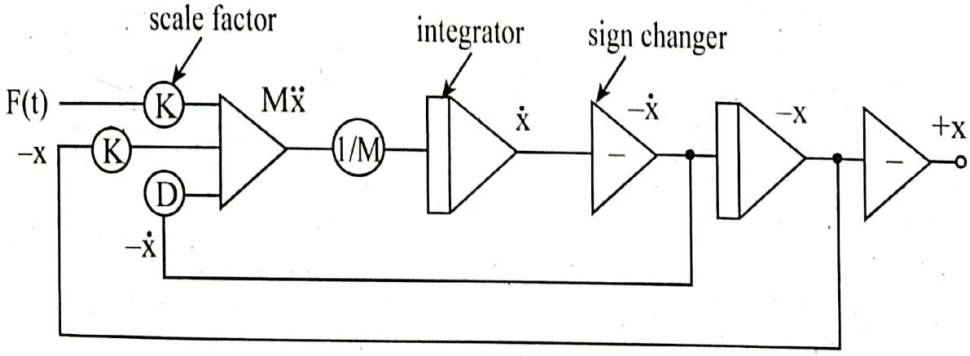


Fig.: Analog computer model for the automobile suspension problem [1]

3.5.2 Analog Computer Model of Liver

Consider the process how a chemical thyroxine is removed from the blood stream. When thyroxine is injected into the blood stream, it is carried to the liver. The liver converts thyroxine into iodine

which is absorbed into the bile. Since, conversion and absorption do not occur instantaneously, some thyroxine reenters the blood stream and is again carried to the liver.

A mathematical model can be constructed to illustrate the rate at which thyroxine is transferred from blood vessels, liver, and bile. Thus, our model should have three independent variables.

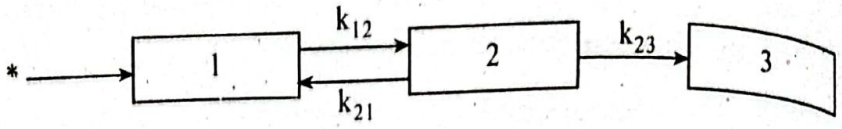


Fig.: Mathematical model of the liver.

In the figure, compartment 1 represents the blood vessels, compartment 2 represents the liver, and compartment 3 represents the bile.

This model results three differential equations which are as follows:

$$\frac{dx_1}{dt} = \dot{x}_1 = -k_{12}\, x_1 + k_{21}\, x_2$$

$$\frac{dx_2}{dt} = \dot{x}_2 = k_{12}\, x_1 - (k_{21} + k_{23})\, x_2$$

$$\frac{dx_3}{dt} = \dot{x}_3 = k_{23}\, x_2$$

When a model has more than one independent variable, a separate block diagram is drawn for each of the independent variables and then interconnected.

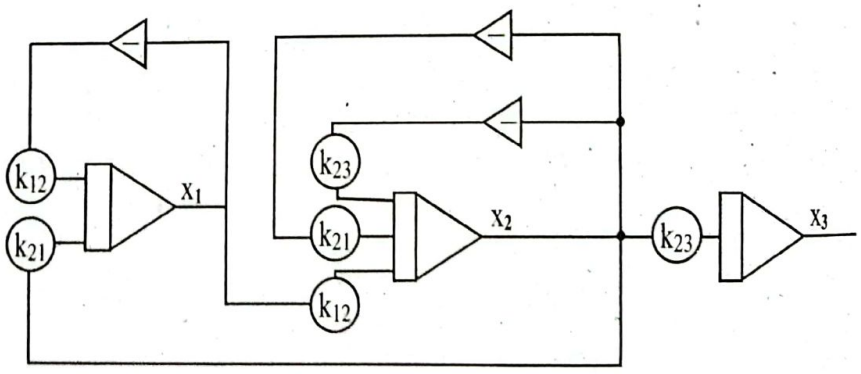


Fig.: Analog computer model of liver [2]

There are three integrators in the model. Each integrator solves the equation for x_1, x_2, and x_3.

The first integrator solves the equation

$$\dot{x}_1 = -k_{12}\, x_1 + k_{21}\, x_2$$

The second integrator solves the equation

$$\dot{x}_2 = k_{12} x_1 - (k_{21} + k_{23})\, x_2$$

The third integrator solves the equation

$$\dot{x}_3 = k_{23} x_2$$

We can analyze the values of x_1, x_2, and x_3 along with their coefficients to understand the flow rate and amount of flow in and from each blocks.

3.6 Hybrid Computers

Hybrid computers are computers that exhibit features of analog computers and digital computers. The digital component normally serves as the controller and provides logical and numerical operations, while the analog component often serves as a solver of differential equations and other mathematically complex problems.

Analog computers are well suited for modeling and simulating systems that exhibit continuous behavior, such as mechanical, electrical, and hydraulic systems. They are fast, simple, and easy to use, but are not very accurate and are not well suited for complex, logical operations.

Digital computers, on the other hand, are very accurate and flexible but are much slower and more complex than analog computers. They are well suited for performing complex logical operations but are not as effective at modeling continuous systems.

Hybrid computers are designed to overcome these limitations by combining the strengths of both analog and digital computers. They are often used in applications where both speed and accuracy are important, such as real-time control systems and scientific simulations.

There are several different types of hybrid computers, including mixed-signal computers, which combine analog and digital circuits on a single chip, and digital-to-analog computers, which use digital circuits to control analog devices.

Hybrid computers are used in several applications such as petrol pumps, electrocardiogram machines, ATMs, control industrial processes, radar systems, artificial satellite, etc.

3.6.1 Hybrid Simulation

If the system under study is of a continuous nature, then analog computer is used for simulation. If the system under study is of a discrete nature, then digital computer is used for simulation. However, there are plenty of scenarios when an analog computer and a digital computer are linked together to provide simulation as in the case of a system where both continuous and discrete subsystems are present. The simulation on such scenario is known as *hybrid simulation*. The term "hybrid simulation" can be applied for any synchronous or asynchronous deployment of continuous simulation and discrete simulation.

3.7 Digital-Analog Simulators (DAS)

To avoid the disadvantages of analog computers, digital computers have *digital-analog simulators*. Digital-analog simulators allow a continuous model to be programmed on a digital computer in essentially the same way as it is solved on an analog computer. The languages contain macro-instructions that carry out the action of adders, integrators, and sign changers. Linking together these macro-instructions is equivalent to connecting the operational amplifiers in analog computers.

Digital analog simulation is a programming technique which makes a digital computer operate much like an analog computer. As an example, consider the application of digital analog simulation (DAS) to programming an IBM 7090 computer. DAS combines amazing ease and speed of programming with high computing speed. The DAS input language allows a simple and

concise description of an analog-style block diagram of the problem to be solved. Also, the construction of a DAS block diagram is more straight-forward than an analog computer block diagram.[3]

Nowadays, more powerful techniques have been developed for continuous simulation using a digital computer. So, digital-analog simulators are not now in extensive use.

3.8 Continuous System Simulation Languages (CSSLs)

Continuous system simulation languages (CSSLs) use familiar statement-type of input for digital computers, and allow a problem to be programmed directly from the equations of a mathematical model, thereby eliminating the need of equations to be broken into functional elements. In addition to simple analog functions such as addition and integration, CSSLs contain a variety of algebraic and logical expressions to describe the relationships between variables.

CSSLs are designed to assist engineers and scientists to mathematically model, analyze, and evaluate the dynamic behavior of physical phenomena. CSSLs are easily learned and applied to many types of problems in all sciences and engineering disciplines. The problems can usually be coded in a short time, executed immediately, and evaluated quickly by inspection of graphic output in several forms.[4]

CSSLs are typically used in combination with computer software tools that can solve the differential equations that represent the system being studied. These tools allow users to simulate the behavior of the system under different conditions and inputs, and to analyze the results in a variety of ways. CSSLs are used in a variety of applications, including the optimization of system design, the prediction of system performance, and the identification of potential problems or failures.

Let us take *Continuous System Modeling Program, Version III (CSMP III)* to best explain CSSLs. A *CSMP III* is an early

computer software designed for modelling and solving differential equations numerically. A CSMP III program is constructed from the following three general types of statements:

- **Structural statements:**

  These statements are FORTRAN-like statements and functional blocks designed for repeat operations that frequently occur in the model definition.

- **Data statements:**

  These statements assign numerical values to parameters, constants, and initial conditions.

- **Control statements:**

  These statements specify options in the assembly and execution of a program and the choice of output of the results.

If a model is represented by an equation

$$Y = 6\,X/W + (Z - 3)^2$$

Then, in CSMP III program, the following statement would be used:

```
Y = 6.0*X/W + (Z - 3.0)**2.0
```

Some CSMP III Functional Blocks

Function	General form
1. Integrator $Y = \int_0^t X\,dt + IC$	Y = INTGRL (IC, X) Y (0) = IC
2. Exponential $Y = e^x$	Y = EXP (X)
3. Natural algorithm $Y = \ln(X)$	Y = ALOG (X)
4. Trigonometric sine $Y = \sin(X)$	Y = SIN(X)

3.9 Feedback System

A *feedback system* is such that uses the output of a process or system as input to regulate or control the process or system, thus introducing a level of dependencies among input and output signals in the system. With the use of feedback in communication systems, satisfactory responses and robust performance can generally be achieved. In a system with feedback, also known as a *closed-loop control system*, the past output may influence the present or future outputs.

There are two main types of feedback systems: *negative feedback systems* and *positive feedback systems*.

In a *negative feedback system*, the output is used to reduce or oppose the input signal. This type of feedback is used to maintain stability and accuracy in the system. For example, the thermostat in a heating system is a negative feedback system that uses the temperature of the room as the input signal and the output is the control signal that adjusts the heating element to maintain the desired temperature.

In a *positive feedback system*, the output is used to amplify or enhance the input signal. This type of feedback can lead to unstable behavior, but can also be used to create oscillating or self-sustaining systems. For example, a microphone and amplifier can form a positive feedback loop, where the microphone converts sound waves into an electrical signal, the amplifier amplifies the signal, and the amplified signal is fed back into the microphone, creating a feedback loop that can result in loud, sustained feedback.

Feedback systems are important because they allow systems to adjust their behavior in response to changing conditions or inputs. They are used in various applications, including control systems, communication systems, and biological systems.

REFERENCES

[1] Geoffrey Gordon. System Simulation. Second edition. Prentice Hall of India Private Limited, 2008, p.63.

[2] Geoffrey Gordon. System Simulation. Second edition. Prentice Hall of India Private Limited, 2008, p.64.

[3] Gaskill et al. DAS – A DIGITAL ANALOG SIMULATOR. 1963.

[4] Dr. Ralph C. Huntsinger. Continuous System Simulation Languages (CSSL's). Proceedings of the 1984 Winter Simulation Conference.

Solution to Some Important Questions

1. Write down the differences between dynamic physical model and dynamic mathematical model.

Ans:

Dynamic physical model	Dynamic mathematical model
A dynamic physical model is a physical representation of a system, such as a physical prototype.	A dynamic mathematical model is a mathematical representation of a system, such as a set of equations.
Dynamic physical models represent the real-world system as closely as possible.	Dynamic mathematical models are more abstract and simplified.
In dynamic physical models, inputs are applied to the physical system, and the outputs are observed directly.	In dynamic mathematical models, inputs are applied to the mathematical equations, and the outputs are computed using the equations.
Dynamic physical models can be expensive to build and maintain.	Dynamic mathematical models can be relatively cheap to develop and maintain.
Changes to the physical system can be time-consuming and expensive to implement.	Changes to a dynamic mathematical model can be made quickly and easily.

Exercise

1. Explain the significance of differential equation in simulation. Explain the analog computer model of the liver with the necessary equation and diagram. [2080 Shrawan, 2074 Bhadra, 2074 Magh]
2. Define analog method. Explain with the example of automobile suspension problem. Discuss the advantages of analog computer over digital computer. [2079 Chaitra]
3. Define hybrid simulation. Explain the analog simulation model of a liver. [2079 Jestha]
4. Why are differential equations important in simulation and modeling? What is an analog computer? Explain the analog method for automobile suspension problem with necessary equations and figures. [2078 Chaitra]
5. Why do we need a digital analog simulator? Explain the analog model of the automobile suspension problem with an equation and diagram. [2077 Chaitra]
6. Differentiate between the dynamic physical and mathematical models with reference to the simulation of automobile wheel suspension. [2076 Bhadra]
7. Define analog computers. Explain analog methods using the model of a liver with the necessary diagram. [2076 Bhadra]
8. What is an analog method? Explain with the example of an automobile suspension problem. [2075 Bhadra, 2073 Bhadra]
9. Design and explain the analog method of automobile suspension problem. Explain the feedback system with a suitable example. [2073 Magh]
10. What is an analog computer? Explain its pros and cons. [2072 Ashwin]
11. What is continuous system simulation? Explain analog computers with practical examples. [2072 Magh]

CHAPTER 4

QUEUING SYSTEM

CHAPTER - 4

QUEUING SYSTEM

4.1 Queuing System

Queuing theory is a branch of mathematics that studies and models the act of waiting in lines, or queues. People at railway ticket windows, vehicles at a petrol pump or at a traffic signal, customers in the bank, products at a machining center, and television sets at a repair shop are a few examples of queues.

A *queuieng system* is a place where *customers* arrive from time to time at a service facility, join a *queue* (waiting line), eventually get served, and finally leave the system. Examples of queuing systems thus can include banks, hospitals, call centers, transportation systems, production systems, repair and maintenance facilities, etc. *Queueing models* (whether analyzed mathematically or through simulation) provide valuable insights for designing and evaluating the performance of queueing systems.

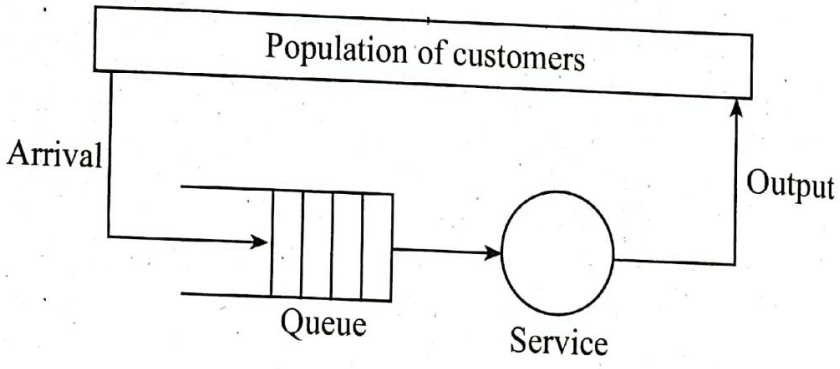


Fig.: Elements of a queuing system.

The main elements of a queueing system are: *customers* and *servers*. The term *customer* refers to anything that arrives at a facility and requires service. Examples of customer can include people, machines, trucks, mechanics, patients, airplanes, e-mail, clothes, etc. The term *server* refers to any resource (person, machine, etc.) that provides the requested service. Examples of server can include washing machines, doctors, receptionists,

mechanics, runways, CPUs in a computer, etc. The service facility may contain more than one server.

Given below are some examples of queuing systems:

System	Servers	Customers
Bank	Tellers	Customers
Hospital	Doctors, Nurses, Beds	Patients
Computer system	CPU, I/O devices	Jobs
Manufacturing systems	Machines, Workers	Parts
Airport	Runways, Gates	Airplanes, Travelers
Communications network	Nodes, Links	Messages, Packets

4.2 Elements of Queuing System

1. Population of Customers

The population of *potential customers (calling population)* may be assumed to be finite or infinite. For example, consider the personal computers of the employees of a small company that are supported by an IT staff of three technicians. When a computer fails, needs new software, etc., it is attended by one of the IT staff. The computers are customers who arrive at the instant they need attention. The IT staff are the servers who provide repairs, software updates, etc. The calling population is finite and consists of the personal computers at the company.

In systems with a large population of potential customers, the calling population is usually assumed to be infinite. Examples of infinite populations include the potential customers of a restaurant, bank, personal computers of the employees of a very large company, etc. Even though the actual population could be finite but large, it is generally safe to use infinite population models.

In an *infinite calling-population model*, the arrival rate does not depend on the number of customers who have left the

calling population and joined the queueing system. *Arrival rate* is defined as the average number of arrivals per unit of time. Whereas in *finite calling-population model*, the arrival rate depends on the number of customers being served and waiting.

2. Arrival

Arrival defines the way that customers enter the system. For infinite-population models, the *arrival process* is usually characterized in terms of interarrival times of successive customers. Arrivals may occur at *scheduled times* or at *random times*. Also, customers may arrive *one at a time* or in *batches* (*constant size* or *random size*).

For arrival occurring at random times, it is necessary to know probability distribution describing the interarrival times. Random arrivals is best modeled by Poisson arrival process. Typical examples of random arrivals include arrival of people to restaurants, arrival of phone calls to a call center, etc.

For scheduled arrivals, it is usually easier to model the positive or negative deviations from the scheduled arrival time, rather than the interarrival times. Typical examples of scheduled arrivals include arrival of patients to a doctor's clinic, arrivals of train at the railway station, etc.

For situations, when at least one customer is assumed to always be present in the queue, the server is never idle because of a lack of customers. A typical example of this is, in a production system in a factory, sufficient raw material (customer) is always available.

For *finite-population models*, the arrival process is characterized as follows. A customer is defined as *pending* when that customer is outside the queueing system and a member of the potential calling population. For example, a hospital patient is pending when they are resting, and becomes not pending the instant they call for the nurse.

3. Queue or Waiting Line

Queue represents a certain number of customers waiting for service. In many queueing systems, there is a limitation to the number of customers that may be in the waiting line or system. No further customers are allowed to enter until space becomes available after the completion of a service. *System capacity* also known as *maximum queue size* is the maximum number of customers that may wait in the queue (plus the one(s) being served).

Queue behavior refers to the actions of customers while in a queue waiting for service to begin. There are different situations, a customer may encounter upon arrival. If a customer upon arrival, sees too long queue, and decides not to enter the system, the customer is said to have *balked*. If a customer stays in the line for some time, sees that the line is moving too slowly, and leaves the line, the customer is said to have *reneged*. If a customer switches from one line to another thinking it has chosen a slow line, the customer is said to have *jockeyed*.

Queue discipline refers to the logical ordering of customers in a queue and determines which customer will be chosen for service when a server becomes free. Following are common queue disciplines:

- First-In-First-Out (FIFO)
- Last-In-First-Out (LIFO)
- Service In Random Order (SIRO)
- Shortest Processing Time First (SPT)
- Service according to Priority (PR)

4. Service

Service represents some activity that takes time and that the customers are waiting for. The service times may be constant or of random duration. Successful models of service

times in different situations are *exponential*, *Weibull*, *gamma*, *lognormal*, and *truncated normal distributions*.

Each service center in a queueing system consists of some number of servers, c, working in parallel. Parallel service mechanisms are either single server (c = 1), multiple servers ($1 < c < \infty$), or unlimited servers ($c = \infty$).

5. Output

Output represents the way customers leave the system. Output is mostly ignored by theoretical models but sometimes the customers leaving the server enter the queue again.

So, the characteristics of queuing system can be explained under the following headings:

- **The calling population**
- **The arrival process**
- **Queue behavior and queue discipline**
- **Service times and the service mechanism**

4.3 Model of Queueing System

4.3.1 Queueing System

A *queueing system* is a mathematical model used to study the behavior of a system that receives incoming requests or customers and processes them in a specific order. Queueing systems are commonly used to model systems such as call centers, banks, hospitals, computer systems, etc.

Queuing models are used to describe the behavior of the queuing system which consists of the queue and the server. A key goal of a queueing system model is to understand the behavior of the system under different conditions and to make predictions about performance metrics such as the average wait time for customers, the utilization of system resources, and the probability of system overload.

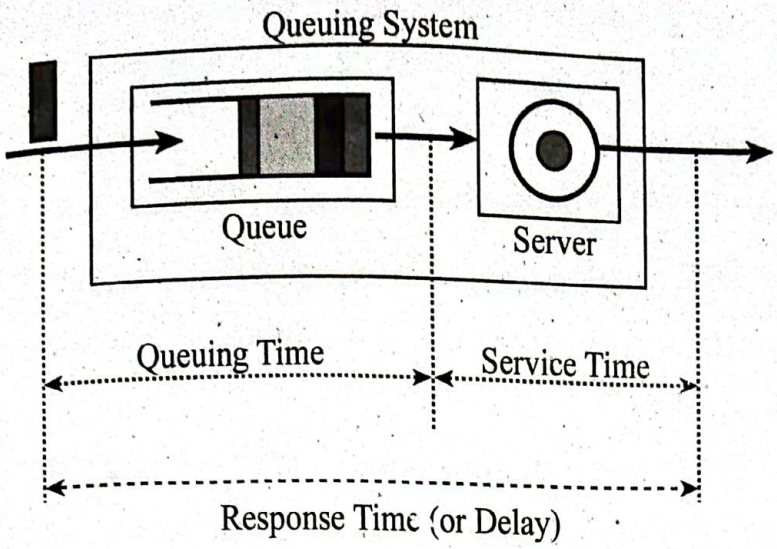


Fig.: Model of queueing system

Queuing models are built with a certain arrival rate (λ) and service rate (μ) and those parameters are used for evaluating system performance and whether the customer has to wait for the service being idle or not.

4.3.2 Components of a Queueing System Model

The key components of a queueing system model are:

- **Arrival process:**
  Arrival process describes the flow of customers arriving at the system and is typically modeled using a probability distribution such as the Poisson or exponential distribution.
- **Service process:**
  Service process describes how customers are serviced by the system and is typically modeled using a probability distribution such as the exponential or deterministic distribution.
- **Queue discipline:**
  Queue discipline describes the order in which customers are serviced.
- **Number of servers:**
  This describes the number of resources (e.g., employees, machines) available to service customers.

Based on the above components, there are multiple models developed to represent the queueing system, some of the famous models are:

- M/M/1 (Markovian arrival, Markovian service, one server)
- M/M/C (Markovian arrival, Markovian service, C servers)
- M/M/C/K (Markovian arrival, Markovian service, C servers, K-capacity)
- G/G/1 (General arrival, general service, one server)
- G/G/C (General arrival, general service, C servers), etc.

Once the model is built, various performance metrics such as mean waiting time, mean queue length, utilization, etc. can be calculated or approximated using mathematical analysis or numerical methods. With the help of these metrics, the system can be evaluated and optimized to improve performance and reduce customer waiting times.

4.4 Types of Queueing System

There are several types of queueing systems, each with its characteristics and behaviors. The most common types can be described with the combination of different numbers of servers and queue. Commonly single channel single server queue is used for modeling basic systems. If there are multiple servers or queue, then it can be categorized in one of the following groups:

1. Single Queue Single Server

In *single queue single server* system, customers form a single queuing line and there is only one server to process the requests. It is the simplest form of queuing system. Once the customer arrives at the facility, depending upon the server utility, it either waits in queue or directly gets the service. If the server is idle, then the customer is eligible for the service otherwise, they have to join the queue. Some examples of single queue single server are ATM line for a

particular bank, printing stations, I/O system in a computer, etc.

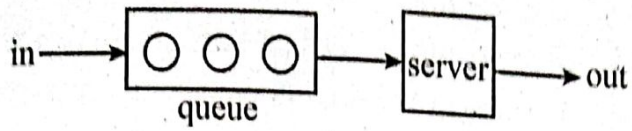


Fig.: Single queue single server

The total server utilization (ρ) is given by the equation:

$\rho = \lambda/\mu$

Where,

λ is customer arrival rate

μ is customer service rate

2. Single Queue Multiple Server

In *single queue multiple server* system, several identical service providers are available, and customers or entities wait in a queue while being served by one of the available servers. Typically, the serving discipline is *First in First Out (FIFO)*. It is assumed that the total population and queue length is infinite and the servers are serving to the infinite customers. Examples of this system are airline ticket check-in counter, post office line, government service line, etc.

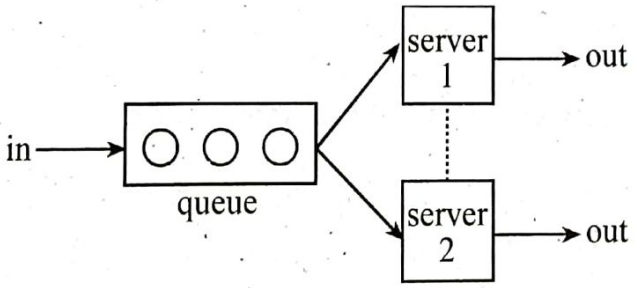


Fig.: Single queue multiple server

The total server utilization (ρ) is given by the equation:

$\rho = \lambda / c\mu$

where

λ is customer arrival rate

μ is customer service rate

c is the number of servers

3. Multiple Queue Multiple Server

In *multiple queue multiple server* system, the customers have a choice to join one of either servers given that the servers they require are not idle. The servers are responsible for same or different task and will serve their respective queue. The average server utilization cannot be determined easily as other kinds of systems because of its complex behaviors such as choice of queue for customers if the servers are performing different kind of jobs, one or multiple servers being idle for too long, customers switching the queue, and so on. Examples of multiple queue multi server include service (call) centers, hospital emergency room, Internet data centers, etc.

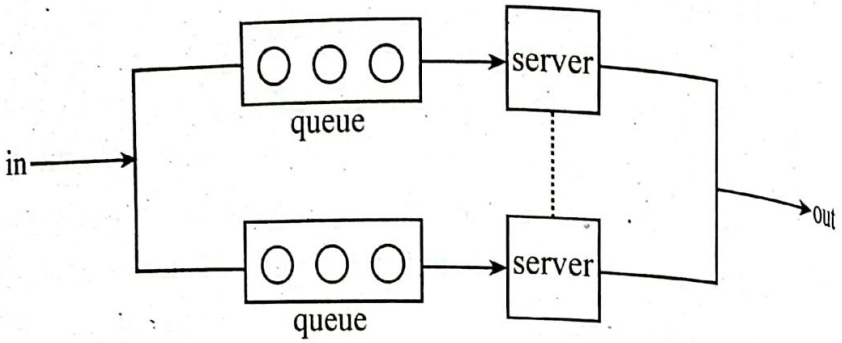


Fig.: Multiple queue multiple server

4.5 Queuing Notation

4.5.1 Kendall Notation: 1/ 2/ 3(/4 /5/ 6)

The Kendall notation, also known as Kendall's notation or the Kendall-Hunt notation, is a way of describing the characteristics of a queueing system using a compact notation using combination of characters and numbers. The notation is named after David George Kendall, who developed it in the 1950s. The notation consists of a set of symbols that describe the properties of the system, such as the number of servers, the queue discipline, and the arrival process.

The Kendall notation is a powerful tool for classifying and analyzing queueing systems, as it allows us to easily identify the key properties of a system and compare it to other systems. It is

important to note that the Kendall notation only gives a quick overview of the system, but not all the details of it.

The most comprehensive notation has the following format:

A/B/s/q/c/p

- **A** represents the *interarrival-time distribution* i.e., arrival pattern. The common symbols for A are **M** (Markov or exponential distribution), **D** (deterministic arrival with constant service duration), $\mathbf{E_k}$ (Erlang distribution), **G** (general (any) distribution), and **GI** (general (any) distribution with independent random values).
- **B** stands for the *service-time distribution* i.e., service pattern. The common symbols for A are **M** (Markov or exponential distribution), **D** (deterministic arrival with constant service duration), $\mathbf{E_k}$ (Erlang distribution), **G** (general (any) distribution), and **GI** (general (any) distribution with independent random values).
- **s** denotes the number of servers.
- **q** is the queueing discipline (FIFO, LIFO, FCFS, LCFS, SIRO). This letter is omitted for FIFO or if not specified.
- **c** means queue capacity. This letter is omitted for unlimited queues.
- **p** represents the size of calling population. This letter is omitted for unlimited size.

Examples:

- **D/M/1:** Deterministic input, one exponential server, one unlimited FIFO or unspecified queue, unlimited customer population.
- **M/G/5/32:** Poisson input, five servers with general distribution, system capacity 32, unlimited customer population.
- **M/D/6/10/FIFO:** Markovian (Poisson's) arrival, six deterministic servers, queue is orderly queue with maximum size 10, unlimited customer population.

4.6 Measurement of System Performance

The main long-run measures of performance of queueing systems are:

- Long-run time-average number of customers in the system (L)
- Long-run time-average number of customers in the queue (L_q).
- Long-run average time spent in system (W) per customer.
- Long-run average time spent in the queue (W_q) per customer.
- Server utilization (ρ) – *Server utilization* is defined as the proportion of time that a server is busy.

The knowledge of the average number of customers in the system or in the queue helps to determine the space requirements of the waiting entities. The knowledge of the average waiting time in the queue is necessary for determining the cost of waiting in the queue. Also, too long waiting line may discourage the prospectus customers, while no queue may suggest that the service offered is not of good quality to attract customers.

Let's look at a single server queue and determine its system performance.

4.6.1 Single Server Queue (M/M/1)

M/M/1 stands for single waiting line with Poisson's (Markovian) arrival (distribution) and a single exponential server i.e., one server with exponential distribution.

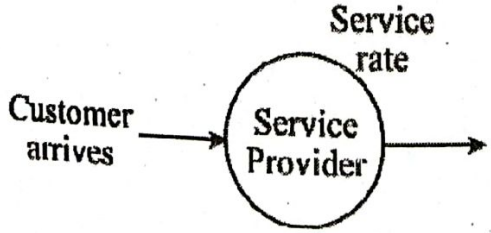


Fig.: Block diagram of M/M/1 queue

The following figure represents M/M/1 queue (single server model).

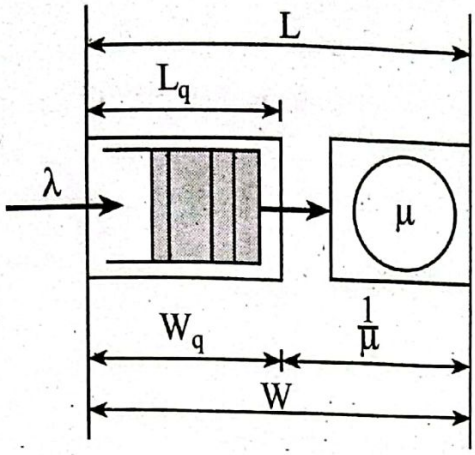


Fig.: M/M/1 queue model

The customer arrival rate i.e., demand is denoted by λ, and the average service rate i.e., capacity is denoted by μ.

Let, λ denotes customer arrival rate and μ represents customer service rate.

Then, *system utilization* (ρ) can be calculated as:

$$\rho = \frac{\lambda}{\mu} \text{ for single server model}$$

Note: $\rho = \frac{\lambda}{n\mu}$ for 'n' server model

Average number of customers in the service system, L is calculated as:

$$L = \lambda/(\mu - \lambda)$$

Average number of customers in waiting line i.e. queue, L_q is given as:

$$L_q = \rho \times L$$

Average time spent in the system, including service, W is obtained as

$$W = 1/(\mu - \lambda)$$

Average time spent in the waiting line, W_q

$$W_q = \rho \times W$$

- Probability that there are n customers in the system (i.e, queueing and also being service), P_n is given as:

  $P_n = \rho^n(1 - \rho)$

- Probability of zero customers (idle time), P_0

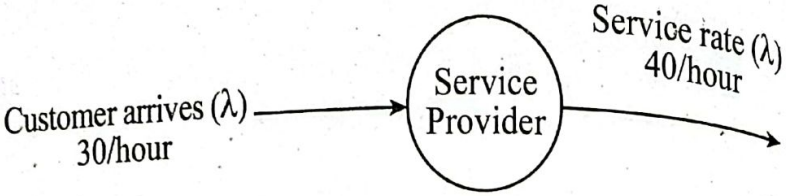
  $P_0 = 1 - (\lambda/\mu)$

- Little's Law

  $\lambda W = L$

Example:

Consider the following waiting line.



Here,

Demand = Customer arrival rate (λ) = 30 customers/hr

Capacity = Average service rate (μ) = 40 customers/hr

System utilization (ρ) = λ/μ = 30/40 = 0.75 i.e. 75%

It means 45 min/hr. is busy, 15 min/hr. is idle time.

Customers in service system (L) $= \lambda/(\mu - \lambda)$

$= 30/(40 - 30)$

$= 3$ i.e., 3 customers

Customers in queue (L_q) $= \rho \times L$

$= 0.75 \times 3$

$= 2.25$

∴ Average number of customers waiting = 2.25

Time spent in the system (W) $= 1/(\mu - \lambda)$

$= 1/(40 - 30)$

$= 1/10$

$= 6$ i.e., 6 minutes

Time spent in waiting line $(W_q) = \rho \times W$

$= 0.75 \times 6$

$= 4.5$ i.e., 4.5 minutes.

$\therefore$ Average waiting time = 4.5 minutes

Average service time = $1/\mu$

$= 1/40$

$= 0.025$

i.e., 0.025 hours or 1.5 minutes.

Hence, the performance of the system could be better if $\lambda = \mu$, which is ideal and denotes, the system is busy serving customers with no idle time. But when $\lambda < 1$, there is more idle time in the waiting line.

4.7 Network of Queues

A *network of queues* refers to a system where multiple queues are connected in a network configuration and a customer departing from one queue may be routed to another. Queues can be linked together to form a network of queues which reflect the flow of customers through a number of different service stations.

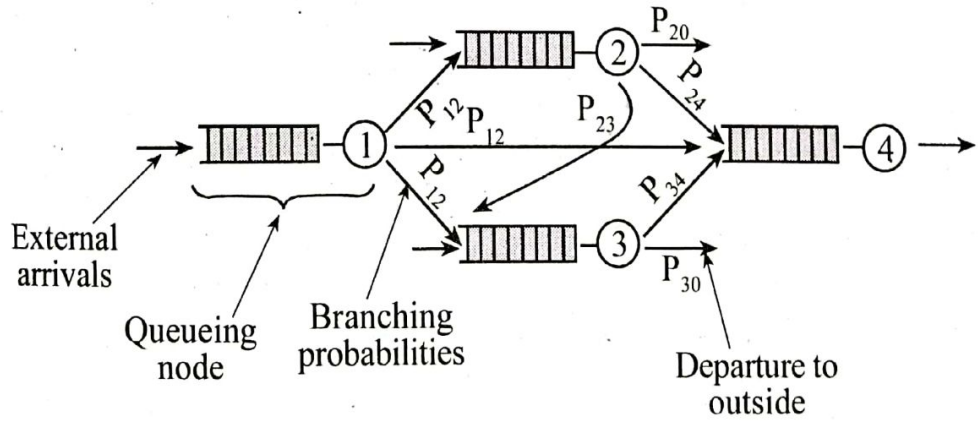


Fig.: Network of queues

One of the most common ways to model a network of queues is to use a mathematical model that represents the network of queues as a collection of interconnected nodes, where each node represents a queue. The interconnections between nodes are represented by

arcs, and the flow of customers through the network is represented by the movement of customers along the arcs.

There are different types of queueing network models, including open network models, closed network models, and mixed network models. A brief description of these models is given below:

1. Open Network Model

Open network model is a network of queues where customers enter the system at one or more source nodes and leave the system at one or more sink nodes. Example: Packet switched data network.

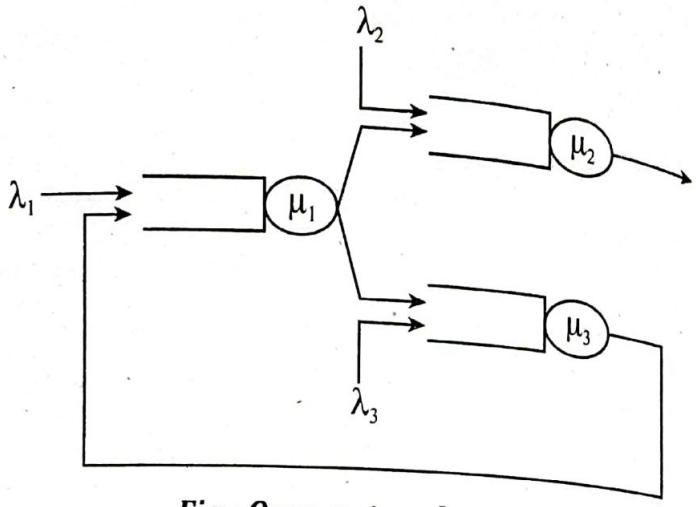


Fig.: Open network model

2. Closed Network Model

Closed network model is a network of queues where customers enter the system at one or more source nodes and eventually cycle through the system, returning to the source nodes. Fixed number of customers (K) are trapped in the system and circulate among the queues. Example: CPU job scheduling problem.

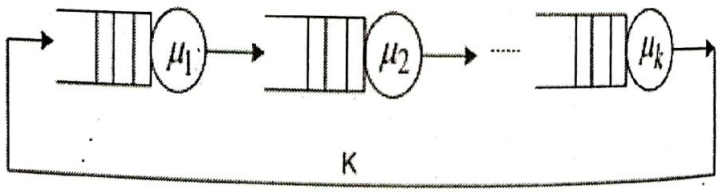


Fig.: Closed network model

3. Mixed Network Model

Mixed network model is a network of queues that combines elements of both open and closed network models. Example: simple model of virtual circuit (VC) that is window flow controlled.

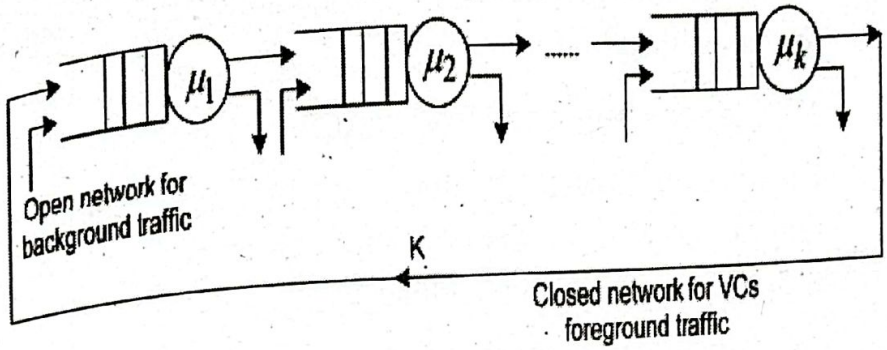


Fig.: Mixed network model

4.8 Applications of Queuing System

Queuing system are used almost in every sector ranging from hospitals to computer systems. Here are some of the most common application areas of queuing system:

1. Telecommunications and Networking

Network traffic management: Queuing systems help manage the flow of data packets in computer networks, ensuring efficient utilization of network resources.

Call centers: Queuing systems manage incoming calls, distributing them to available agents based on various factors such as priority and agent availability.

2. Transportation and Logistics

Traffic management: Queuing models are used to analyze traffic patterns and optimize traffic flow at intersections, toll booths, and other traffic congestion points.

Airport security and checkpoints: Queuing models help optimize the allocation of security resources to minimize wait times for passengers.

3. **Service Industry**

Retail stores: Queuing systems optimize checkout lines in supermarkets and retail stores, improving customer satisfaction and reducing waiting times.

Healthcare: Queuing models are used in hospitals and clinics to manage patient flow, appointment scheduling, and resource allocation.

4. **Manufacturing and Production**

Production lines: Queuing systems help analyze and optimize production line efficiency, ensuring smooth flow of materials and reducing bottlenecks.

Inventory management: Queuing models assist in determining reorder points and optimal inventory levels to meet demand while minimizing holding costs.

5. **Information Technology**

Server management: Queuing systems help optimize server allocation and resource utilization in data centers and cloud computing environments.

Job scheduling: Queuing models are used in batch processing and job scheduling, ensuring efficient utilization of computing resources.

6. **Financial Services**

Bank queues: Queuing systems help banks manage customer queues, optimize teller assignments, and reduce waiting times.

ATM networks: Queuing models are used to optimize the placement and maintenance of ATMs to meet customer demand.

7. **Public Services**

Government offices: Queuing systems help manage queues for services like obtaining permits, licenses, and other government-related tasks.

Post offices: Queuing models assist in optimizing the allocation of postal service resources to minimize customer wait times.

8. **Entertainment and Hospitality**

Theme parks: Queuing systems help design ride queues to enhance visitor experience and manage crowd flow efficiently.

Restaurants: Queuing systems assist in managing table reservations and walk-in customers to optimize seating arrangements.

9. **Emergency Services**

Emergency rooms: Queuing models help hospitals manage patient triage and resource allocation during peak demand periods.

These are just a few examples of how queuing systems are applied to various real-world situations to optimize resource allocation, minimize wait times, enhance customer satisfaction, and improve overall operational efficiency.

1. Attendants in a workshop manage the tool cribs as mechanics, assumed to be from an infinite calling population, arrive for service. Assume Poisson arrivals at the rate of 2 mechanics per minute and exponentially distributed service times with a mean of 40 seconds. Calculate the optimum number of attendants required and the probability of having zero mechanics in the system. [2079 Jestha]

Solution:

Arrival rate (λ) = 2 per minute

Mean service time $= \frac{1}{\mu} = 40$ seconds

Service rate (μ) $= \frac{1}{40}$ seconds

$= \frac{60}{40}$ per minute

$= \frac{3}{2} = 1.5$ per minute

Here, the offered load is greater than 1.

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

$$= \frac{2}{3/2} = \frac{4}{3} > 1$$

So, more than one server is needed if the system is to have a statistical equilibrium.

Thus at least c = 2 attendants are needed. (multi-server)

i.e. optimum number of attendants required = 2

Probability of having zero mechanics in the system (P_0) =?

For single-server,

$$P_0 = 1 - \rho$$

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

But, for multiple server queues,

$$P_0 = \left\{ \left[\sum_{n=0}^{c-1} \frac{\left(\frac{\lambda}{\mu}\right)^n}{n!} \right] + \left[\left(\frac{\lambda}{\mu}\right)^c \left(\frac{1}{c!}\right) \left(\frac{c\mu}{c\mu - \lambda}\right) \right] \right\}^{-1}$$

$$= \left\{ \left[\sum_{n=0}^{2-1} \frac{\left(\frac{4}{3}\right)^n}{n!} \right] + \left[\left(\frac{4}{3}\right)^2 \left(\frac{1}{2!}\right) \left(\frac{2 \times \frac{3}{2}}{2 \times \frac{3}{2} - 2}\right) \right] \right\}^{-1}$$

$$= \left\{ 1 + \frac{4}{3} + \left(\frac{16}{9}\right) \times \left(\frac{1}{2}\right) \times (3) \right\}^{-1} = 0.2$$

---

2. Define the meaning of:

i. **M/D/8/15/LIFO** [2071 Bhadra]

ii. **D/M/1/FIFO/20/510**

iii. **M/D/2/60/150/FIFO** [2073 Magh]

iv. **M/M/4/20/200/FCFS** [2074 Bhadra]

v. **D/M/2/LIFO/18** [2074 Bhadra]

vi. **M/D/8/15/1000/LIFO** [2075 Bhadra]

vii. **M/Em/2/9/70/SIRO** [2076 Bhadra]

viii. **D/M/6/FIFO/25/3000** [2077 Chaitra]

ix. **D/M/1/LIFO/20/80** [2078 Chaitra]

x. **D/M/1/LIFO/10/50** [2080 Shrawan]

Solution:

- **M/D/8/15/LIFO** [2071 Bhadra]

  Single waiting line with Poisson's or Markovian arrival, deterministic service distribution with 8 servers, queue is stack with maximum size 15 (0 to 14), unlimited customer population.

- **D/M/1/FIFO/20/510**

  According to Kendall classification (notation), the above queuing system represents the system with deterministic input (arrival), a single server with exponential distribution, queue is orderly queue (FIFO) with system capacity 20, the population of customers is 510.

- **M/D/2/60/150/FIFO** [2073 Magh]

  According to Kendall classification (notation), the above queuing system represents the system with Poisson's or Markovian arrival, deterministic service distribution with 2 servers, queue is orderly queue (FIFO) with system capacity 60, the population of customers is 150.

- **M/M/4/20/200/FCFS** [2074 Bhadra]

  According to Kendall classification (notation), the above queuing system represents the system with Poisson's or Markovian arrival, 4 servers with exponential distribution, FCFS (First Come First Serve) system with capacity 20, the population of customers is 200.

- **D/M/2/LIFO/18** [2074 Bhadra]

  According to Kendall classification (notation), the above queuing system represents the system with deterministic input (arrival), 2 servers with exponential distribution, queue is stack (LIFO) with system capacity 18, unlimited customer population.

- **M/D/8/15/1000/LIFO** [2075 Bhadra]

  Single waiting line with Poisson's or Markovian arrival, deterministic service distribution with 8 servers, queue is stack with maximum size 15 (0 to 14), the population of customers is 1000.

- **M/Em/2/9/70/SIRO** [2076 Bhadra]

  According to Kendall classification (notation), the above queuing system represents the system with Poisson's or

Markovian arrival, 2 servers with Erlang distribution, SIRO (Serve in Random Order) system with capacity 9, the population of customers is 70.

- **D/M/6/FIFO/25/3000** [2077 Chaitra]

According to Kendall classification (notation), the above queuing system represents the system with deterministic input (arrival), 6 servers with exponential distribution, queue is orderly queue (FIFO) with system capacity 25, the population of customers is 3000.

- **D/M/1/LIFO/20/80** [2078 Chaitra]

According to Kendall classification (notation), the above queuing system represents the system with deterministic input (arrival), single server with exponential distribution, queue is stack (LIFO) with system capacity 20, the population of customers is 80.

- **D/M/1/LIFO/10/50** [2080 Shrawan]

According to Kendall classification (notation), the above queuing system represents the system with deterministic input (arrival), single server with exponential distribution, queue is stack (LIFO) with system capacity 10, the population of customers is 50.

Exercise

1. Discuss the characteristics of queuing system? What do you mean by Kendall notation in queuing system? [2080 Shrawan, 2075 Bhadra]
2. Explain the characteristics of queueing system. [2079 Chaitra]
3. Define queueing system. Explain Kendal's notation. [2079 Jestha]
4. Discuss the elements of the queuing system and its applications. [2078 Chaitra]
5. Discuss the characteristics and application of queuing system. [2077 Chaitra]
6. What are the elements of a queuing system? What are the characteristics of queuing system? [2076 Bhadra]
7. Define queuing system with a block diagram and its uses. [2074 Bhadra]
8. Explain the queuing notation with at least two examples and also explain the possibility of default values for notation. Explain about different elements of the queue system. [2074 Magh]
9. What is a queuing system? List the various characteristics of queuing system. Explain the role of queuing system in simulation study. Explain Kendall's notation with an example. [2073 Bhadra]
10. What is queuing model, explain with the figure. Explain the Kendall notation with an example. [2073 Magh]
11. Discuss any one practical application of queuing system. [2078 Ashwin]
12. How can you measure the system performance of queuing system? Explain. [2071 Magh]

CHAPTER 5

MARKOV CHAIN

5.1 Markov Process

5.2 Key Features of Markov Chain

5.3 Application of Markov Chain

CHAPTER - 5

MARKOV CHAIN

5.1 Markov Process

A *Markov process* is a random process that changes over time and whose future behavior is independent of the past and depends only on its current state. Markov processes, named after Andrei Markov, are among the most important of all random processes.

A *Markov state*, refers to a specific configuration or condition of the system that is characterized by certain variables or properties. It represents a snapshot or instantaneous description of the system at a given point in time.

A *Markov chain* is a type of Markov process, it's a discrete-time stochastic process that satisfies the Markov property. It consists of a sequence of random states, with the property that no matter how the system arrived at its current state, the possible future states are fixed. The system is memoryless, meaning that the probability of being in a particular state at a future time step depends only on the current state, and not on the history of how the system arrived at that state.

A Markov chain can be represented as a directed graph, called a *state diagram*, where each node represents a state and each directed edge represents a transition between states. The edges of the graph are labeled with the transition probabilities, which are the probabilities of transitioning from one state to another. The time spent by the job in such a queue is the Markov process and the number of jobs in the queue is a Markov chain.

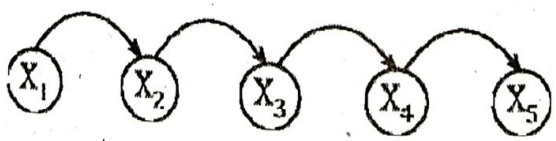


Fig.: Simple Markov chain

Transition Matrix

In a Markov chain, the transition matrix is a square matrix that represents the probabilities of transitioning from one state to another. It provides a complete description of the dynamics of the Markov chain. The transition matrix is usually denoted as P and has dimensions n×n, where n is the number of states in the Markov chain. Each element P(i, j) of the matrix represents the probability of transitioning from state i to state j in a single step.

The elements of the transition matrix satisfy the following properties:

1. **Non-negativity:**

   Each element P(i, j) is non-negative, indicating a probability value between 0 and 1.

2. **Row sum property:**

   The sum of each row of the matrix is equal to 1, as the probabilities of transitioning to different states must sum up to 1.

The transition matrix allows us to calculate the probabilities of transitioning between states over multiple steps. By taking powers of the transition matrix, we can determine the probabilities of being in different states after a certain number of steps.

The transition matrix is a fundamental component of Markov chain analysis and is used to study various properties of the chain, such as steady-state probabilities, expected hitting times, and absorbing states.

Example:

$$P = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.2 & 0.7 \end{bmatrix}$$

In this example, the Markov chain has three states: state 1, state 2, and state 3. The transition matrix P shows the probabilities of transitioning from one state to another.

For instance, P(1, 2) = 0.3 represents the probability of transitioning from state 1 to state 2 in a single step, which is 0.3 or 30%. Similarly, P(2, 1) = 0.2 represents the probability of transitioning from state 2 to state 1 in a single step, which is 0.2 or 20%.

The row sums of the matrix add up to 1, as required by the row sum property. For example, the row sum for the first row is 0.4 + 0.3 + 0.3 = 1.

By raising the transition matrix to higher powers, we can compute the probabilities of transitioning between states over multiple steps. For example, P^2 represents the probabilities after taking two steps, P^3 represents the probabilities after three steps, and so on.

5.2 Key Features of Markov Chain

A Markov chain is a sequence of random experiments that satisfies the following conditions:

- The experiments have a finite or countable set of possible outcomes, which are considered as discrete states.
- The outcome of each experiment depends only on the current state and is independent of any previous or future states.
- The probabilities of transitioning from one state to another remain fixed and do not change over time or between consecutive transitions.

Example of a Markov Process

Consider the following weather scenario.

- **If it rains today, there is 40% probability that it will rain tomorrow and 60% probability of not raining tomorrow.**
- **If it doesn't rain today, there is 20% probability that it will rain tomorrow and 80% probability of not raining tomorrow.**

Calculate:

(a) **What will be the probability if today is not raining, then not rain the day after tomorrow?**

(b) **What is the probability if today is not raining, then rain the day after tomorrow?**

(c) **What will be the probability if today is not raining, then rain after three days?**

Solution:

Representing above condition in stochastic finite state machine (FSM)

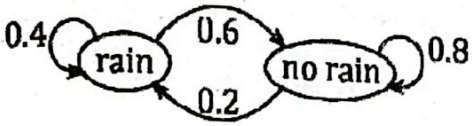


Fig.: Finite state machine

Transition matrix for above condition is:

$$P = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}$$

Let's interpret the transition matrix:

- Row 1 represents the probabilities when it is currently raining today:
  - **P(1, 1) = 0.4:** The probability of raining tomorrow given that it is currently raining today is 0.4 or 40%.
  - **P(1, 2) = 0.6:** The probability of not raining tomorrow given that it is currently raining today is 0.6 or 60%.
- Row 2 represents the probabilities when it is currently not raining today:
  - **P(2, 1) = 0.2:** The probability of raining tomorrow given that it is currently not raining today is 0.2 or 20%.
  - **P(2, 2) = 0.8:** The probability of not raining tomorrow given that it is currently not raining today is 0.8 or 80%.

$$\text{Here, } P = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}$$

$$\text{Now, } P^2 = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}$$

$$= \begin{bmatrix} 0.4\times0.4+0.6\times0.2 & 0.4\times0.6+0.6\times0.8 \\ 0.2\times0.4+0.8\times0.2 & 0.2\times0.6+0.8\times0.8 \end{bmatrix}$$

$$= \begin{bmatrix} 0.28 & 0.72 \\ 0.24 & 0.76 \end{bmatrix}$$

$$\text{Again, } P^3 = \begin{bmatrix} 0.28 & 0.72 \\ 0.24 & 0.76 \end{bmatrix}\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{bmatrix}$$

$$= \begin{bmatrix} 0.28\times0.4+0.72\times0.2 & 0.28\times0.6+0.72\times0.8 \\ 0.24\times0.4+0.76\times0.2 & 0.24\times0.6+0.76\times0.8 \end{bmatrix}$$

$$= \begin{bmatrix} 0.256 & 0.744 \\ 0.248 & 0.752 \end{bmatrix}$$

a. $P^2(2, 2) = 0.76$

b. $P^2(2, 1) = 0.24$

c. $P^3(2, 1) = 0.248$

5.3 Application of Markov Chain

Markov chains are used to analyze trends and predict the future. Markov chains have various applications in different fields. Some of these are:

1. **Finance and Economics:**

   Markov chains are used in modeling and predicting financial markets, stock prices, and economic systems. They can be used to analyze the transition probabilities between different market states and assess the risk associated with investment portfolios.

2. **Natural Language Processing:**

Markov chains are used in language modeling and text generation. They can be used to generate realistic and coherent sentences based on the statistical properties of a given text corpus. Markov models are also used in speech recognition and machine translation.

3. **Biology and Genetics:**

Markov chains are used in modeling biological systems, such as DNA sequences, protein folding, and population dynamics. They can help understand the transition probabilities between different genetic states and simulate the evolution of biological systems over time.

4. **Queueing Theory:**

Markov chains are extensively used in modeling and analyzing queueing systems, such as customer queues in service centers, traffic flow, and network congestion. They can provide insights into system performance metrics like waiting times, service rates, and resource utilization.

5. **Artificial Intelligence and Reinforcement Learning:**

Markov decision processes (MDPs) are a variant of Markov chains used in reinforcement learning. MDPs model sequential decision-making problems and are used to develop optimal policies for agents in dynamic environments.

6. **Epidemiology:**

Markov chains are used in modeling the spread of infectious diseases. They can help assess the probabilities of individuals transitioning between different health states (e.g., susceptible, infected, recovered) and predict the future progression of the epidemic.

These are just a few examples, and Markov chains have applications in many other fields, including engineering, physics, social sciences, and operations research. They provide a powerful

framework for analyzing systems with probabilistic transitions and are widely used for modeling and prediction purposes.

Internet Applications of Markov Chain

A Markov chain is a mathematical model that represents a system with a sequence of states and the probabilities of transitioning between those states. In the context of Internet applications, Markov chains are widely used to analyze and model various aspects of online behavior.

One prominent application of Markov Chains in the online world is the PageRank algorithm, developed by Sergey Brin and Larry Page for Google's search engine. The PageRank algorithm assigns a numerical value, known as the PageRank score (PR), to each webpage in a network of interconnected pages. The score represents the importance or relevance of a webpage based on the probability of a random surfer navigating to that page.

The PageRank algorithm uses a Markov chain model, where each webpage corresponds to a state, and the links between webpages represent the transitions between states. The transition probability from one webpage to another is determined by factors such as the number of incoming links to a webpage (k_i), the total number of known webpages (N), and a damping factor (α) that accounts for the likelihood of a random surfer teleporting to any webpage. The transition probability can be calculated as follows:

$$P(\text{transition}) = \alpha/k_i + (1 - \alpha)/N \text{ (for all linked webpages)}$$
$$P(\text{transition}) = (1 - \alpha)/N \text{ (for all pages which are not linked)}$$

The parameter α is typically set to around 0.85, although it can be adjusted based on specific requirements.

Markov chains are also employed to analyze user behavior and navigation patterns on websites. By constructing a Markov chain based on user interactions, such as clicks and page visits, it is possible to model the likelihood of users moving from one page to another. The transition probabilities in this case can be estimated based on historical data or user preferences.

Furthermore, Markov chains can be utilized in online advertising and recommendation systems. By modeling user preferences and the transitions between different products or content items, Markov Chains can help predict user behavior and make targeted recommendations. The transition probabilities in this case can be based on user preferences, click-through rates, or other relevant factors.

Overall, Markov chains provide a powerful framework for understanding and modeling various aspects of internet applications. From webpage ranking and user behavior analysis to personalization and recommendation systems, Markov Chains play a crucial role in understanding and optimizing online experiences.

1. Given that a chance of a Honda user to buy a Honda in the next purchase is 60% and that his next purchase will be Yamaha or Bajaj is 30% and 10% respectively. Also, if the chance of a Yamaha user to buy a Yamaha in their next purchase is 70% and that his next purchase will be Bajaj or Honda is 10% and 20% respectively and a chance of a Bajaj user to buy Bajaj in next purchase is 65% and that his next purchase will be Honda or Yamaha is 20% and 15% respectively.

a. Design the Markov Model for the given scenario.

b. What is the probability to buy Honda after two purchases for a current Yamaha user? [2078 Chaitra]

Solution:

a.

The finite state machine (Markov model) of the above scenario is

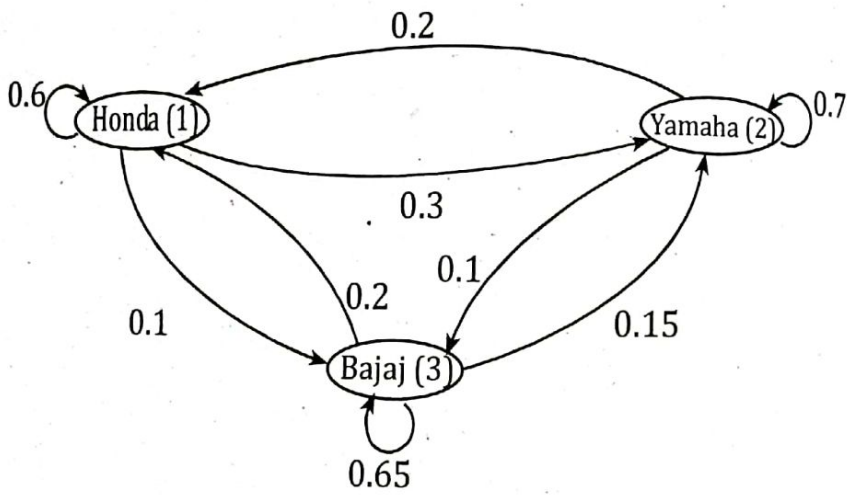


Fig.: Stochastic FSM

The transition matrix, P is given by:

$$P = \begin{bmatrix} 0.60 & 0.30 & 0.10 \\ 0.20 & 0.70 & 0.10 \\ 0.20 & 0.15 & 0.65 \end{bmatrix}$$

b. In order to find the transition matrix after two purchases, we square the above matrix P.

$$P^2 = \begin{bmatrix} 0.60 & 0.30 & 0.10 \\ 0.20 & 0.70 & 0.10 \\ 0.20 & 0.15 & 0.65 \end{bmatrix}^2 = \begin{bmatrix} 0.44 & 0.405 & 0.155 \\ 0.28 & 0.565 & 0.155 \\ 0.28 & 0.2625 & 0.4575 \end{bmatrix}$$

Therefore, the probability of a current Yamaha user buying Honda after two purchases, $P^2(2, 1) = 0.28$ or 28%.

2. Given that a chance of a Momo lover to order Momo in their next purchase is 70% and that his next purchase will be Burger is 30% and the chance of a Burger lover to order a Burger at the next purchase is 80% and the chance that his next purchase will be Momo is 20%.

a. What is the probability to buy a Burger after three purchases of a current Momo lover?

b. If 55% of people order Momo today and 45% of people order Burgers, what percentage of people will use Momo after 3 purchases? **[2076 Bhadra]**

Solution:

a.

The finite state machine (Markov Model) of the above scenario is

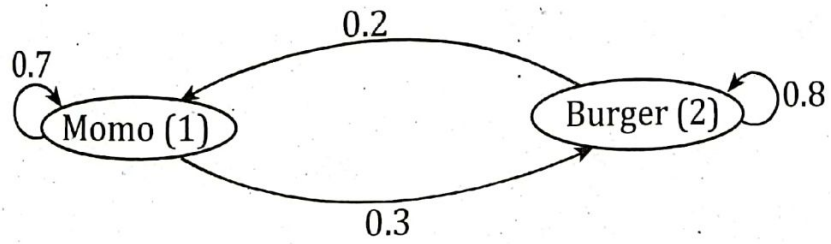


Fig.: Stochastic FSM

The transition matrix, P is given by:

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix}$$

The transition matrix after three purchases is given by:

$$P^3 = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix}^3 = \begin{bmatrix} 0.475 & 0.525 \\ 0.35 & 0.65 \end{bmatrix}$$

Probability of a current Momo lover buying a Burger after a third purchase, $P^3(1,2) = 0.525$

b.

If 55% of people order Momo today, then to calculate the percentage of the user that will order Momo after 3 purchases, we have

$Q_0 = [0.55 \quad 0.45]$

Now, $Q_3 = Q_0 \times P^3$

$$= [0.55, 0.45] \times \begin{bmatrix} 0.475 & 0.525 \\ 0.35 & 0.625 \end{bmatrix}$$

$$= [0.4187 \quad 0.5812]$$

Therefore, 41.87% of people will order Momo after 3 purchases.

3. Given that a chance of a Ford car user to buy a Ford car in their next purchase is 70% and that his next purchase will be a Scorpio car is 30% and the chance of a Scorpio car user buying a Scorpio car at the next purchase is 80% and the chance that his next purchase will be ford car is 20%. What is the probability to buy a Scorpio car after three purchases of a current Ford car user? If 70% of user uses a Ford car today, what percentage of the user will use Scorpio after 3 purchases? [2075 Bhadra]

Solution

The finite state machine (Markov Model) of the above scenario is

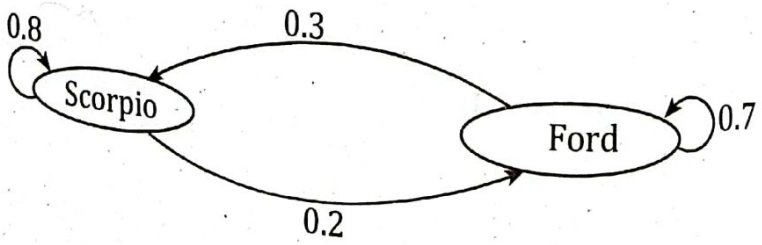


Fig.: Stochastic FSM

The transition matrix, P is given by:

$$P = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$$

The transition matrix after three purchases is given by:

$$P^3 = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}^3 = \begin{bmatrix} 0.65 & 0.35 \\ 0.525 & 0.475 \end{bmatrix}$$

If 70% of user uses Ford car today, then to calculate the percentage of the user that will use Scorpio after 3 purchases, we have:

$$Q_0 = [0.3 \quad 0.7]$$

Now, $Q_3 = Q_0 \times P^3$

$$= [0.3 \quad 0.7] \times \begin{bmatrix} 0.65 & 0.35 \\ 0.525 & 0.475 \end{bmatrix}$$

$$= [0.5625 \quad 0.4375]$$

Therefore, 56.25% of the user will use Scorpio after 3 purchases.

4. Given that the chance of a Sony user to buy Sony at the next purchase is 90% and that his next purchase will be Samsung is 10% and the chance of a Samsung user to buy Samsung at his next purchase is 85% and the chance that his next purchase will be Sony is 15%. What is the probability to buy Sony after three purchases of a current Samsung user? If 60% of user uses Sony today, what percentage of user uses Samsung after three purchases? [2074 Bhadra]

Solution:

The finite state machine (Markov model) of the above scenario is

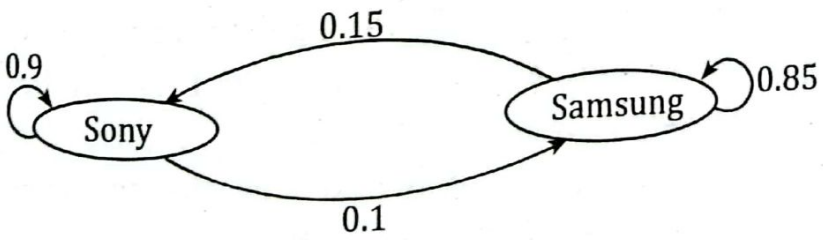


Fig.: Stochastic FSM

The transition matrix, P is given by:

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.15 & 0.85 \end{bmatrix}$$

The transition matrix after three purchases is given by:

$$P^3 = \begin{bmatrix} 0.9 & 0.1 \\ 0.15 & 0.85 \end{bmatrix}^3 = \begin{bmatrix} 0.768 & 0.231 \\ 0.346 & 0.6531 \end{bmatrix}$$

Probability of a current Samsung user buying Sony in the third purchase = 0.346

If 60% of people use Sony today, then to calculate the percentage of the user that will use Samsung after 3 purchases, we have

$Q_0 = [0.6, 0.4]$

Now, $Q_3 = Q_0 \times P^3$

$$= [0.6 \quad 0.4] \times \begin{bmatrix} 0.768 & 0.231 \\ 0.346 & 0.6531 \end{bmatrix}$$

$$= [0.599 \quad 0.399]$$

Therefore, 39.9% of people will use Samsung after 3 purchases.

---

5. Design a Markov model and transition matrix for given data. Answer the following questions based on the model.

If a person purchases coke now the probability of purchase of coke next time is 80%.

If a person purchases Pepsi now the probability of purchasing Pepsi next time is 70%.

Problems:

a. **Find the probability of using coke for a current Pepsi user in 4^{th} purchase.**
b. **Find the probability of a person using Pepsi now for a current Pepsi purchaser after four purchases.**
c. **A population of 5250 (out of 10200) in a study of soft drink user who uses coke now, find the population of purchasing Pepsi in 4^{th} purchase.** [2074 Magh]

Solution:

a) The finite state machine (Markov Model) of the above scenario is

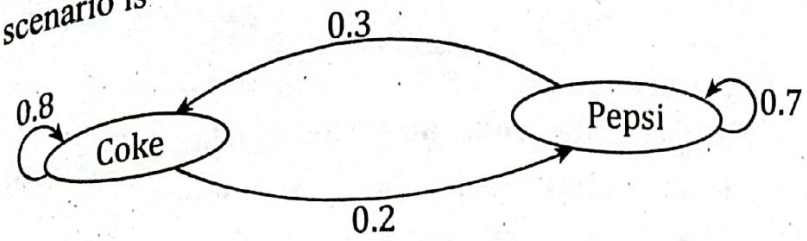


Fig.: Stochastic FSM

The transition matrix, P is given by:

$$P = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$$

The transition matrix after four purchases is given by:

$$P^4 = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}^4 = \begin{bmatrix} 0.625 & 0.375 \\ 0.562 & 0.437 \end{bmatrix}$$

Probability of the current Pepsi purchaser buying Coke in the fourth purchase = 0.562

b)

Probability of the current Pepsi purchaser buying Pepsi in the fourth purchase = 0.437

c)

If 51.47% (5250/10200) of people use Coke today, then to calculate the percentage of the user that will use Pepsi after 4 purchases, we have

$$Q_0 = [0.5147 \quad 0.4853]$$

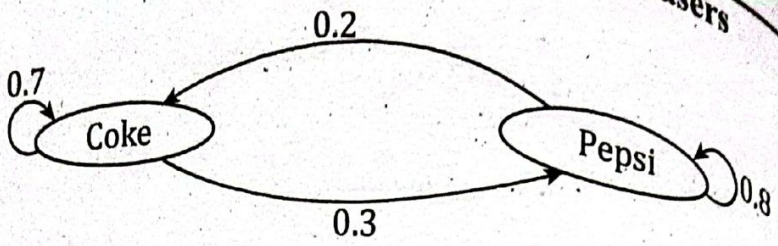
Now, $Q_3 = Q_0 \times P^4$

$$= [0.5147 \quad 0.4853] \times \begin{bmatrix} 0.625 & 0.375 \\ 0.562 & 0.437 \end{bmatrix}$$

$$= [0.594 \quad 0.405]$$

Therefore, 40.5% of people will use Pepsi after 4 purchases.

6. Given figure shows Coke and Pepsi purchasers



a. If currently coke purchaser, what is the probability of Pepsi purchasing in 3rd purchase?

b. If 55% of people use Coke today, what percentage of people will use Coke after 3 purchases? [2071 Bhadra]

Solution:

a)

The finite state machine (Markov model) of the above scenario is

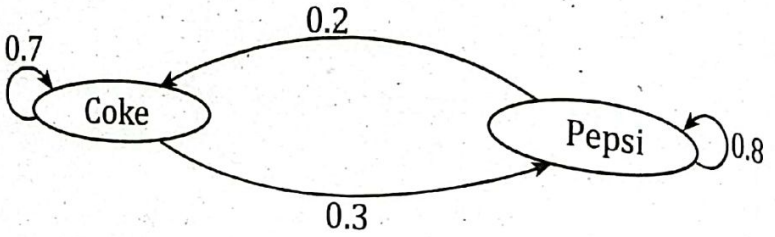


Fig.: Stochastic FSM

The transition matrix, P is given by:

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix}$$

The transition matrix after three purchases is given by:

$$P^3 = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix}^3$$

$$= \begin{bmatrix} 0.475 & 0.525 \\ 0.35 & 0.65 \end{bmatrix}$$

Probability of the current Coke purchaser buying Pepsi in the third purchase = 0.525

b)

If 55% of people use Coke today, then to calculate the percentage of the user that will use Coke after 3 purchases, we have

$Q_0 = [0.55 \quad 0.45]$

Now, $Q_3 = Q_0 \times P^3$

$$= [0.55 \quad 0.45] \times \begin{bmatrix} 0.475 & 0.525 \\ 0.35 & 0.625 \end{bmatrix}$$

$$= [0.4187 \quad 0.5812]$$

Therefore, 41.87% of people will use Coke after 3 purchases.

Exercise

1. Explain the key features of Markov chain and also explain the application of Markov chain with a suitable example. [2080 Shrawan, 2078 Jestha]
2. Explain Markov chain with example. [2078 Chaitra]
3. Explain the Markov chain process, its features, and its application with examples. [2077 Chaitra]
4. What is Markov chain? Find the transition probability of a Pepsi user who will buy Coke on a third purchase. The transition probability is as below:

	Pepsi	**Coke**
Pepsi	0.7	0.3
Coke	0.1	0.9

5. What do you mean by Markov chain? Explain with Internet applications. [2072 Magh] [2071 Magh]

CHAPTER 6

RANDOM NUMBER

CHAPTER - 6

RANDOM NUMBER

Random numbers are a key element of simulation models of almost all discrete systems, as they are used to represent uncertainty and randomness in the system being simulated. These days, most computer languages have a subroutine, object, or function that will generate a random number. As the name suggests, the value of random numbers cannot be predicted. Random numbers are the samples drawn from uniformly distributed random variables between some satisfied intervals which have an equal probability of occurrence. Such numbers are always independent and uniformly distributed, so there exists no correlation between successive numbers. *Random number table* is a table of numbers generated in an unpredictable, haphazard (hit-or-miss) that are uniformly distributed within certain interval.

Random numbers are typically generated using a random number generator (RNG), which is an algorithm that produces a sequence of numbers that are statistically random.

There are two types of random number generators, each with its characteristics and strengths:

- **Pseudorandom number generators (PRNGs):**

  These are deterministic algorithms that generate a sequence of numbers that appears random, but are determined by a seed value. PRNGs are fast and easy to use, but the sequence of numbers they produce is ultimately predictable.

- **True random number generators (TRNGs):**

  These generate random numbers using physical processes, such as atmospheric noise or radioactive decay. TRNGs are more secure than PRNGs, but they may be slower and more expensive to use.

  In simulation, is important to use a suitable random number generator that produces numbers that are random enough for

the specific application. This can help to ensure that the simulation results are accurate and reliable.

6.1 Properties of Random Numbers

A sequence of random numbers R_1, R_2, R_3, ..., must have following two important properties:

- **Uniformity:**

  Random numbers should be distributed uniformly, meaning that they have equal probability of occurring. This is important because it helps to ensure that the simulation results are not biased toward certain values. For example, if the random numbers are not uniformly distributed, then the simulation results may be skewed towards certain values, which could lead to incorrect conclusions being drawn.

- **Independence:**

  Random numbers should be independent, meaning that the current value of a random variable has no relation with the previous values. This is important because it helps to ensure that the simulation results are not correlated or dependent on one another. For example, if the random numbers are not truly independent, then the simulation results may be correlated, which could lead to incorrect conclusions being drawn.

- The probability density function (pdf) of a random number R_i is given by:

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

- The expected value of each R_i is given by:

$$E(R) = \int_0^1 x\,dx = \left[\frac{x^2}{2}\right]_0^1 = \frac{1}{2}$$

- The variance of each R_i is given by:

$$V(R) = \int_0^1 x^2\,dx - [E(R)]^2 = \left[\frac{x^3}{3}\right]_0^1 - \left(\frac{1}{2}\right)^2 = \frac{1}{12}$$

6.2 Types of Random Numbers

Following are the types of random numbers:

i. True Random Number

True random numbers are generated from physical processes like the decay of a radioactive material.

ii. Pseudo Random Number

Pseudo-random numbers or *false random numbers* are those numbers that appear to be statistically random, despite having been produced by a completely deterministic and repeatable arithmetic operations or formulas. These numbers are not truly random since the arithmetic operation is known and the sequence of random numbers can be repeated. However, these numbers very closely mimic the properties of random numbers.

iii. Quasi Random Number

Quasi (virtual) random numbers are not designed to appear random, rather to be uniformly distributed. One aim of such numbers is to reduce and control errors in Monte Carlo simulations.

6.3 Generation of Pseudo-Random Numbers

There are several methods for generating pseudo-random numbers, including linear congruential generators, multiplicative generators, combined linear congruential generators, etc. These methods all involve starting with a seed value and then using a mathematical formula to generate a sequence of numbers that appears random. The quality of the pseudo-random numbers generated by these methods can vary, and it is important to choose a method that generates numbers with the desired properties.

During the generation of pseudo-random numbers, certain errors or problems could occur leading to departures from ideal randomness.

Some of these are as follows:

- The generated numbers may exhibit a non-uniform distribution.
- The generated numbers may take on discrete values rather than continuous values.
- The mean of the generated numbers may be too high or too low.
- The variance of the generated numbers may be too high or too low.
- There may be dependencies among the generated numbers, such as auto-correlation between numbers, sequential patterns of being successively higher or lower than adjacent numbers, and occurrence of several numbers above the mean followed by several numbers below the mean.

Thus, before employing a pseudo-random number generator, it should be properly validated, by testing the generated random numbers for randomness.

When selecting a PRNG for use in a simulation, there are several important factors to consider. Some of these factors are listed below:

- **The quality of the pseudo-random numbers**:

  It is important to choose a generator that produces numbers that are statistically close to truly random numbers, with a uniform distribution and low correlations between numbers.

- **The period of the generator:**

  The period of a random number generator is the number of numbers it can generate before repeating the same sequence. A longer period is generally desired, as it reduces the likelihood of repeating the same sequence of numbers during the simulation.

- **The speed of the generator:**
  The speed of the generator is important if the simulation requires a large number of random numbers to be generated in a short amount of time. Some generators are faster than others, so it is important to choose a generator that can meet the performance requirements of the simulation.
- **The size of the state:**
  Some generators require a large amount of memory to store their internal state, which can be a problem if the simulation is running on a system with limited memory. It is important to choose a generator that has a reasonable state size for the intended use.
- **The availability of a high-quality implementation:**
  It is important to choose a generator that has a well-tested and reliable implementation that is widely available. This can help to ensure that the generator will behave as expected and will be easy to use in the simulation.

6.4 Random Number Generation Methods

In computer simulation, where a very large number of random numbers is generally required, the random numbers could be obtained by the following methods:

- **Electronic devices:**
  Electronic devices such as hardware random number generators can be used to generate random numbers. These devices use physical phenomena such as noise or radioactive decay to generate random numbers. These devices are generally considered to be the most reliable method for generating random numbers, but they can be expensive.
- **Using tables:**
  Random numbers can be obtained from random number tables stored in the memory of the computer. These tables typically contain a large number of random numbers that

have been generated using a variety of methods, such as electronic devices or mathematical algorithms. The main advantage of using tables is that they are readily available and do not require any special equipment. However, the quality of the random numbers generated from tables may not be as high as those generated using other methods.

Using arithmetic operations:

Random numbers can also be generated using arithmetic operations such as linear congruential generators or multiplicative congruential generators. These methods use mathematical algorithms to generate random numbers based on a starting value or seed. The main advantage of using arithmetic operations is that they are relatively simple and easy to implement. However, the quality of the random numbers generated by these methods may not be as high as those generated using other methods.

6.4.1 Techniques for Generating Random Numbers

1. Linear Congruential Method (Congruence or Residue Method)

The *linear congruential method* is a pseudorandom number generator algorithm commonly used to generate a sequence of numbers that appear random. The method is a type of recurrence relation, which means that each number in the sequence is determined by a formula that depends on the previous number.

In linear congruential method, a sequence of integers X_1, X_2,.. between 0 and m−1 are generated using the formula

$$X_{i+1} = (aX_i + c) \bmod m \quad \text{.....(i)}$$

where, i = 0, 1, 2, 3, ...

The initial value X_0 is called the *seed*, a is called the *multiplier*, c is the *increment*, and m is the *modulus*.

- **Case-I:**
  If $c \neq 0$, then the form is known as *mixed congruenti*
  method.
- **Case-II:**
  If $c = 0$, then the form is known as *multiplicati*
  congruential method.
- **Case-III:**
  If $a = 1$, then the form is known as *additive congruentia*
  method.

To start the sequence, an initial value X_0 is chosen. The firs
number in the sequence X_1 is then calculated using the formula
above (equation i). The second number in the sequence X_2 is ther
calculated using the formula with X_1 as the current value, and so
on. The random numbers R_i is calculated using

$$R_i = \frac{X_i}{m}$$

The linear congruential method has several parameters that can be
adjusted to control the properties of the generated numbers. For
example, changing the value of the modulus m, will affect the
range of the generated numbers: Changing the value of the
multiplier a, will affect the distribution of the generated numbers.

One disadvantage of the linear congruential method is that it can
produce patterns in the generated numbers if the parameters are not
chosen carefully. However, by choosing appropriate values for the
parameters, it is possible to generate pseudo-random numbers that
have good statistical properties.

Analysis of Different Cases

i. Mixed Congruential Method ($c \neq 0$)

Then, equation (i) will remain unchanged.

$X_{i+1} = (aX_i + c) \bmod m$

This recursive relation is a *mixed congruential method* since
increment and multiplier aid in producing the random
integer value.

For example, let $a = 17, \quad m = 100, X_0 = 27, c = 43$

Then, the sequence of X_i and subsequent R_i are calculated as:

$X_1 = (17 \times 27 + 43) \% 100$

$= 502 \% 100$

$= 2$

And $R_1 = X_1/m$

$= 2/100$

$= 0.02$

Similarly,

$X_2 = (17 \times 2 + 43) \% 100 = 77$

$R_2 = 0.77$

$X_3 = (17 \times 77 + 43) \% 100 = 52$

$R_3 = 0.52$

and so on.

Hence, random integer values $X_i = 27, 2, 77, 52, \ldots$

Random numbers $R_i = 0.27, 0.02, 0.77, 0.52, \ldots$

ii. Multiplicative Congruential Method (c = 0)

The equation (i) will be reduced to

$X_{i+1} = (aX_i) \bmod m$

In the above equation, since there is no increment but only a multiplier a; the recursive relation is said to be a *multiplicative congruential method.*

For example, let $a = 17, m = 100, X_0 = 27$

Then, the sequence of X_i and subsequent R_i are calculated as:

$X_1 = (17 \times 27) \% 100$

$= 459 \% 100$

$= 59$

And $R_1 = X_1/m$

$= 59/100$

$= 0.59$

Similarly,

$X_2 = (17 \times 59) \% 100 = 3$

$R_2 = 0.03$

$X_3 = (17 \times 3) \% 100 = 51$

$R_3 = 0.51$

and so on.

Hence, random integer values $X_i = 27, 59, 3, 51, \ldots$

Random numbers $R_i = 0.27, 0.59, 0.03, 0.51, \ldots$

iii. Additive Congruential Method (a=1)

The equation (i) will be then

$X_{i+1} = (X_i + c) \bmod m$

This recurrence relation is called an *additive congruential method.*

For example, let $m = 100$, $X_0 = 27$, $c = 43$

Then, the sequence of X_i and subsequent R_i are calculated as:

$X_1 = (27 + 43) \% 100$

$= 70 \% 100$

$= 70$

And $R_1 = X_1/m$

$= 70/100$

$= 0.70$

Similarly,

$X_2 = (70 + 43) \% 100 = 13$

$R_2 = 0.13$

$X_3 = (13 + 43) \% 100 = 56$

$R(3) = 0.56$

and so on.

Hence, random integer values $X_i = 27, 70, 13, 56, \ldots$

Random numbers $R_i = 0.27, 0.70, 0.13, 0.56, \ldots$

2. **Arithmetic Congruential Method**

In this method, random numbers are generated by the equation

$$X_{i+1} = (X_{i-1} + X_i) \bmod m$$

For example, let $X_1 = 9$, $X_2 = 13$, $m = 17$

$X_3 = (X_1 + X_2) \bmod m = (9 + 13) \% 17 = 5$

$X_4 = (X_2 + X_3) \bmod m = (13 + 5) \% 17 = 1$

and so on.

The ultimate test of linear congruential generator is how closely the generated numbers R_1, R_2, R_3, ... approximate *uniformity* and *independence*. It also includes secondary properties such as *maximum density* and *period*.

Requirements for a good random number generator:

For a random number generator to be useful in simulation, following requirements should be satisfied.

- The sequence of R.N. must follow the uniform distribution.
- The sequence of R.N. must be statistically independent.
- The sequence of R.N. generated must be reproducible which allows replication of the simulation experiments.
- The sequence must be non-repeating for any desired length.
- Generation of R.N. must be fast because in simulation a large number of random numbers are required. A slow generation greatly increases the time and cost of simulation studies/experiments.
- The technique used in generating R.N. should require little computer memory.

6.5 Test for Random Numbers

The prime properties for random numbers are: *uniformity* and *independence.*

For testing *uniformity*, following frequency tests can be applied:

- **Kolmogorov-Smirnov Test**
- **Chi-Square Test**

For testing *independence*, following tests can be applied:

- **Autocorrelation Test**
- **Gap Test**
- **Poker Test**
- **Runs Test**

6.5.1 Kolmogorov-Smirnov Test

This test is based on the null hypothesis that there is no significant difference between the sample distribution and the theoretical distribution. This test is applicable to any sample size.

Let F(x) be the continuous cumulative distribution function (cdf) of the uniform distribution and $S_N(x)$ be the empirical cdf of the sample of N observations.

From the definition, we have

$$F(x) = x, \quad 0 \le x \le 1$$

and $S_N(x)$ can be expressed as

$$S_N(x) = \frac{\text{number of } R_1, R_2, \ldots\ldots, R_N \text{ which are} \le x}{N}$$

where R_1, R_2,....., R_N is the sample from the random-number generator.

The Kolmogorov-Smirnov test works by calculating largest absolute deviation (D) between F(x) and $S_N(x)$ i.e.,

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

The algorithm of this test includes the following steps:

1. Set up the hypothesis as

   H_0: $R \sim U[0, 1]$

   H_1: $R \nsim U[0, 1]$

2. Rank the data from smallest to largest. If R_i is the i^{th} smallest data, then

   $R_1 \leq R_2 \leq R_3 \leq \ldots\ldots\ldots \leq R_N$

3. Compute D^+ and D^-

   $$D^+ = \max_{1 \leq i \leq N} \left\{ \frac{i}{N} - R_i \right\}$$

   $$D^- = \max_{1 \leq i \leq N} \left\{ R_i - \frac{i-1}{N} \right\}$$

4. Calculate D as

   $D = \max (D^+, D^-)$

5. Note critical value D_α from the table for given α (level of significance) and N (sample size).

6. If calculated value of D of sample is greater than D_α, then the null hypothesis (H_0) that there is no significant difference between the sample distribution and the theoretical distribution is rejected. If calculated D of sample is less than or equal to D_α, then the null hypothesis is not rejected, that is, conclusion can be made that there is no significant difference between the sample distribution (R_1, R_2, ..., R_N) and the uniform distribution.

Comparison between K-S Test and Chi-Square Test

	K-S Test		Chi-Square Test
i.	This test is only appropriate for testing data against continuous distribution.	i.	This test can be used for testing against both continuous and discrete distributions.

	K-S Test		Chi-Square Test
ii.	Applicable for all sample size.	ii.	Applicable to large sample size, $N \geq 50$.
iii.	Differences between observed and expected cumulative probabilities (CDF) is calculated.	iii.	Differences between observed and hypothesized probabilities (PDFs or PMFs) is calculated.
iv.	This test uses each observation in the sample without any grouping. There is better use of data. Cell size is not a problem.	iv.	This test groups observation into a small number of cells. Cell sizes affect the conclusion.

6.5.2 Chi-Square Test

This test is also based on the null hypothesis that there is no significant difference between the sample distribution and the theoretical distribution. This test is applicable to only large sample, that is, $N \geq 50$.

This text applies the sample statistic:

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where O_i = observed number in the i^{th} class

E_i = expected number in the i^{th} class $= \frac{N}{n}$

n = number of classes

Algorithm:

1. Set up the hypothesis as

   H_0: $R \sim U[0, 1]$

   H_1: $R \nsim U[0, 1]$

2. Compute χ_0^2 as

$$\chi_0^2 = \frac{(O_i - E_i)^2}{E_i}$$

$i = 1$ to n

3. Get the value of $\chi^2_{\alpha,df}$ from the table

Where

df (degree of freedom) = n – 1

α = level of significance

4. Compare calculated value of χ_0^2 with critical value $\chi^2_{\alpha,df}$

If $\chi_0^2 \leq \chi^2_{\alpha,df}$, H_0 is not rejected i.e., numbers are uniformly distributed between [0, 1].

Else, $\chi_0^2 > \chi^2_{\alpha,df}$, H_0 is rejected i.e., no evidence of uniformity has been detected.

6.5.3 Autocorrelation Test

Autocorrelation test calculates the correlation between numbers and compares the sample correlation to the expected correlation of zero. This test checks the dependence between numbers in a sequence. For example, consider the following sequence of numbers:

0.14 0.03 0.25 0.19 <u>0.91</u> 0.33 0.66 0.19 0.85 <u>0.95</u>

0.21 0.17 0.35 0.37 <u>0.93</u> 0.43 0.62 0.29 0.77 <u>0.90</u>

0.70 0.51 0.07 0.45 <u>0.97</u> 0.60 0.21 0.38 0.71 <u>0.89</u>

Inspecting this, we see that 5th, 10th, 15th, and so on, has a very large number in that position. So, these numbers might be related. To test if any relationship exists between numbers in the sequence is done through autocorrelation test.

In this test, autocorrelation between every m numbers (m is also known as lag), starting with the i^{th} number. Thus, autocorrelation

(ρ_{im}) between R_i, R_{i+m}, R_{i+2m}, ..., $R_{i+(M+1)m}$ is calculated. Here, M is the largest integer such that $i+(M+1)m \leq N$, and N is the total numbers in the sequence.

Autocorrelation test is also based on the null hypothesis that there is no dependence between numbers in the sample i.e., $\rho_{im} = 0$.

For this, Z_0 is calculated as

$$Z_0 = \frac{\hat{\rho}_{im}}{\sigma_{\hat{\rho}_{im}}}$$

where $\hat{\rho}_{im}$ is the estimator of ρ_{im} and $\sigma_{\hat{\rho}_{im}}$ is the standard deviation of the estimator.

Schmidt and Taylor [1970] has given the formula for $\hat{\rho}_{im}$ and $\sigma_{\hat{\rho}_{im}}$ as

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

$$\sigma_{\hat{\rho}_{im}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

Null hypothesis is rejected if $-z_{\alpha/2} \leq Z_0 \leq z_{\alpha/2}$ where $z_{\alpha/2}$ can be noted from the table.

6.5.4 Gap Test

Gap test is a test for determining whether random numbers are independently distributed or not. The gap test is used to determine the significance of the interval between the recurrences of the same digit. There is a certain gap of length (say x) between the recurrences of some specified digit. Consider the following example:

4, 1, 3, 5, 1, 7, 2, 8, 2, 0, 7, 9, 1, 3, 5, 2, 7, 9, 4, 1, 6, 3, 3, 9, 6, 3, 4, 8, 2, 3, 1, 9, 4, 4, 6, 8, 4, 1, 3

There are seven 3's in the given sequence. So, only six gaps can occur. The first gap has length 10, the second gap is of length 7, the third gap has length 0, and so on.

The probability of a particular gap length can be determined by Bernoulli trial.

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

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

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

The theoretical frequency distribution for randomly ordered digits is given by

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

Algorithm:

1. Set up a hypothesis for testing independence as

   H_0 : R ~ independently distributed i.e $D_\alpha > D$

   H_1 : R $\nsim$ independently

2. Specify the cdf for theoretical frequency distribution given by

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

3. Arrange the observed sample of gaps in a cumulative distribution within specified class intervals with specific interval widths.

4. Calculate the observed frequency distribution $S_N(x)$ as

   $S_N(x)$ = (number of gaps <= x)/total no. of gaps

   = Frequency/total no. of gaps

   where,

   total no. of gaps = total no. of digits – no. of distinct digits.

   $S_N(x)$ is cumulative relative frequency distribution.

5. Calculate D, the maximum deviation between theoretical frequency distribution, F(x) and observed frequency distribution, $S_N(x)$ as

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

6. Determine the critical value i.e. D_α, from Kolmogorov-Smirnov (K-S) table for a specified value of α and sample size N.

7. Compare calculated D and critical value D_α.

   If $D_\alpha > D$, then the null hypothesis of independence is accepted, and the distribution is independent.

   Else, reject the null hypothesis, and the distribution is not independent.

6.5.5 Poker Test

The *Poker Test*, also known as the *Poker-Squared Test*, is a statistical test used to determine whether a sequence of random numbers is independent. The test is based on the concept of a poker hand, which is a set of five cards in a deck of 52 cards.

The Poker test is the test for independence based on the frequency with which certain digits are repeated in a series of numbers. This test not only tests for the randomness of the sequence of numbers, but also the digits comprising of each of the numbers.

The null hypothesis of the Poker Test is that the sequence of random numbers is independent and generated by a uniform distribution. If this hypothesis is true, then the expected number of each type of poker hand can be calculated based on the number of combinations in a deck of cards. The alternative hypothesis is that the sequence of random numbers is not independent. If the observed number of each type of poker hand differs significantly from the expected number, this would indicate that the random numbers are not independent.

To perform the test, the observed and expected frequencies of each type of poker hand are calculated and used to calculate a chi-

$$E(X_i) = \frac{1}{\lambda}$$

and so, $1/\lambda$ is the mean interarrival time. The objective is to develop a procedure for generating values $X_1, X_2, X_3, \ldots$ that have an exponential distribution.

The process can be summarized in following algorithmic steps:

1. Compute the cdf of the desired random variable X. For the exponential distribution, the cdf is

   $F(x) = 1 - e^{-\lambda x}, x \geq 0$

2. Set $F(X) = R$ on the range of X.

   For the exponential distribution, it becomes $1 - e^{-\lambda X} = R$ on the range $x \geq 0$.

   X is a random variable (with the exponential distribution in this case), so $1 - e^{-\lambda X}$ is also a random variable, R. R has a uniform distribution over the interval [0, 1].

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

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

   or, $e^{-\lambda X} = 1 - R$

   Taking *ln* on both sides,

   $-\lambda X = ln\,(1 - R)$

   $\therefore X = -\frac{1}{\lambda} ln\,(1 - R)$

   Equation above is called a random-variate generator for the exponential distribution.

4. Generate (as needed) uniform random numbers $R_1, R_2, R_3, \ldots$ and compute the desired random variates as

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

   for $i = 1, 2, 3, \ldots$.

squared statistic. The chi-squared statistic is compared to a critical value from a chi-squared distribution with (the number of possible types of poker hands minus one) degrees of freedom. If the calculated statistic is greater than the critical value, the null hypothesis is rejected, indicating that the sequence of random numbers is not independent. We will have a total of 10 possible numbers i.e., 0 to 9 that can be combined in different poker hands.

Calculation of Probability

Consider any 3-digit number, there could be only 3 possibilities.

1. P (all different: ABC) = P(1^{st} digit being unique) × P(2^{nd} digit different from 1^{st}) × P(3^{rd} digit diff. from 1^{st} and 2^{nd})
   = (10/10) × (9/10) × (8/10)
   = 0.72

2. P(all same: AAA) = P(1^{st} digit being unique) × P(2^{nd} digit same as 1^{st}) × P(3^{rd} digit same as 1^{st})
   = (10/10) × (1/10) × (1/10)
   = 0.01

3. P(exactly 1 pair: AAB) = ${}^{3}C_{2}$ (P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit diff. as 1^{st}))
   = ${}^{3}C_{2}$ (0.1×0.9)
   = (3!/2!1!) × 0.09
   = 3 × 0.09
   = 0.27

Considering 4 Poker hands (4 digits numbers)

1. P (all same) = P(1^{st} digit) × P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit same as 1^{st}) × P (4^{th} digit same as 1^{st})
   = (10/10) × (1/10) × (1/10) × (1/10)
   = 0.001

2. P(all 4 digit diff.) = P(1st digit) × P(2nd digit different from 1st) × P(3rd digit diff. from 1st and 2nd) × P (4th digit diff as 1st, 2nd and 3rd)

$= (10/10) \times (9/10) \times (8/10) \times (7/10)$

$= 0.504$

3. P(1 pair) $= {}^4C_2 (1 \times 0.9 \times 0.8 \times 0.1)$

$= 4!/2!2! \times (0.1 \times 0.72)$

$= 0.432$

4. P(2 pair) $= {}^4C_2/2 \times {}^2C_2 (1 \times 0.9 \times 0.1 \times 0.1)$

$= 4!/2!2!2 \times 1 (0.01 \times 0.9)$

$= 3 \times 0.09$ 3 × 0.009

$= 0.27$ 0.027

5. P(3 like digits) $= {}^4C_3 \times (1 \times 0.9 \times 0.1 \times 0.1)$

$= 4 \times (0.01 \times 0.9)$

$= 4 \times 0.09$

$= 0.036$

Considering 5-digit numbers there are 7 possibilities with their probabilities as:

1. P(All different) = P(1st digit) × P(2nd digit different from 1st) × P(3rd digit diff. from 1st and 2nd) × P (4th digit diff as 1st, 2nd and 3rd) × P (5th digit diff as 1st, 2nd, 3rd and 4th)

$= (10/10) \times (9/10) \times (8/10) \times (7/10) \times (6/10)$

$= 0.3024$

2. P(1 Pair / 3 diff digits) $= {}^5C_2 (1 \times 0.9 \times 0.8 \times 0.7 \times 0.1)$

$= {}^5C_2 (0.09 \times 0.56)$

$= 0.5040$

3. P(2 pairs) $= {}^5C_2/2 \times {}^3C_2 (1 \times 0.9 \times 0.8 \times 0.1 \times 0.1)$

$= 5!/2!3!2 \times 3 \times (0.01 \times 0.72)$

$= 10/2 \times 3(0.0072)$

$= 0.108$

4. $P(\text{three of a kind}) = {}^5C_3\,(1 \times 0.9 \times 0.8 \times 0.1 \times 0.1)$

$= 5!/2!3! \times (0.01 \times 0.72)$

$= 0.072$

5. $P(\text{Full House}) = {}^5C_3 \times {}^2C_2\,(1 \times 0.9 \times 0.1 \times 0.1 \times 0.1)$

$= 10 \times 1\,(0.001 \times 0.9)$

$= 0.009$

6. $P(\text{Four of a kind}) = {}^5C_4\,(1 \times 0.9 \times 0.1 \times 0.1 \times 0.1)$

$= 5\,(0.001 \times 0.9)$

$= 0.0045$

7. $P(\text{Five of a kind/all same}) = P(1^{st}\text{ digit}) \times P(2^{nd}\text{ digit same as }1^{st}) \times P(3^{rd}\text{ digit same as }1^{st}) \times P(4^{th}\text{ digit same as }1^{st}) \times P(5^{th}\text{ digit same as }1^{st})$

$= (10/10) \times (1/10) \times (1/10) \times (1/10) \times (1/10)$

$= 0.0001$

The algorithm for the poker test is the same as the chi-square test. Once we calculate the expected frequency using the above method and determine the observed frequency from the given data, perform the chi-square test on the obtained frequencies.

6.5.6 Runs Test

This test checks for the independence of the generated random numbers by testing the runs of numbers above and below the mean. It compares the actual number of runs to the expected number of runs and uses a chi-square test for statistical comparison.

6.6 Random Variate Generation

A random variable is a measurable mapping having some distribution, and a random variate is just a member of the co-

domain of a random variable. A random variate is a particular outcome of a random variable. Random variates are the samples generated from a known distribution i.e., random variable and random variates have an inverse relationship.

Suppose X is a random variable which stands for the outcome of tossing a fair dice. So, X can take value from 1 through 6 with equal probability of 1/6. Now you actually toss a dice and get a number 4. This number is a particular outcome of X, and thus a random variate. If you toss again, you may get another different value.

There are several methods to generate random variate, some of them are explained below as subtopics:

6.6.1 Inverse Transform Sampling

The *inverse-transform technique* can be used to sample from the exponential, uniform, Weibull, and triangular distributions, and from empirical distributions. Additionally, it is the underlying principle for sampling from a wide variety of discrete distributions. The technique is explained below in detail for the exponential distribution, it is the most straightforward, but not always the most efficient, technique.

The exponential distribution has the probability density function (pdf)

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

and the cumulative distribution function (cdf)

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

The parameter λ can be interpreted as the mean number of occurrences per time unit. For example, if interarrival times X_1, X_2, X_3, ... had an exponential distribution with rate λ, then λ could be interpreted as the mean number of arrivals per time unit, or the arrival rate. For any i,

$$E(X_i) = \frac{1}{\lambda}$$

and so, $1/\lambda$ is the mean interarrival time. The objective is to develop a procedure for generating values $X_1, X_2, X_3, \ldots$ that have an exponential distribution.

The process can be summarized in following algorithmic steps:

1. Compute the cdf of the desired random variable X. For the exponential distribution, the cdf is

   $F(x) = 1 - e^{-\lambda x}, x \geq 0$

2. Set $F(X) = R$ on the range of X.

   For the exponential distribution, it becomes $1 - e^{-\lambda X} = R$ on the range $x \geq 0$.

   X is a random variable (with the exponential distribution in this case), so $1 - e^{-\lambda X}$ is also a random variable, R. R has a uniform distribution over the interval [0, 1].

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

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

   or, $e^{-\lambda X} = 1 - R$

   Taking *ln* on both sides,

   $-\lambda X = ln\,(1 - R)$

   $\therefore X = -\frac{1}{\lambda} ln\,(1 - R)$

   Equation above is called a random-variate generator for the exponential distribution.

4. Generate (as needed) uniform random numbers $R_1, R_2, R_3, \ldots$ and compute the desired random variates as

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

   for $i = 1, 2, 3, \ldots$.

6.6.2 Acceptance-Rejection Technique

Acceptance-rejection technique is used particularly when inverse cdf doesn't exist in closed form.

For an example, to generate random variates, $X \sim U[1/4, 1]$, following steps are required:

Step 1: Generate a random number $R \sim U[0, 1]$.

Step 2:

a. If $R \geq \frac{1}{4}$, accept $X = R$, then go to Step 3.

b. If $R < \frac{1}{4}$, reject R, and return to Step 1.

Step 3: If we need another random variate, $X \sim U[1/4, 1]$, repeat from Step 1. If not, stop.

Each time Step 1 is executed, a new random number R must be generated. Step 2(a) is an "acceptance" and Step 2(b) is a "rejection", hence the name "acceptance-rejection technique". The efficiency of "acceptance-rejection technique" depends largely on the ability to minimize the number of rejections.

6.6.3 Convolution Method

The probability distribution of a sum of two or more independent random variables is called a *convolution* of the distributions of the original variables. The convolution method thus refers to adding together two or more random variables to obtain a new random variable with the desired distribution.

The convolution relationship:

Let $Y_i \sim G(y)$ be independent and identically distributed (iid) random variables.

$$X = \sum_{i=1}^{n} Y_i$$

The distribution of X is said to be the convolution of Y.

Some common random variables with convolution:

- Binomial Variable = Σ iid Bernoulli variables
- Negative Binomial = Σ iid Geometric variables
- Erlang Variable = Σ iid Exponential variables
- Normal Variable = Σ iid other Normal variables
- Chi-squared Variable = Σ iid Squared normal variables

Algorithm to generate random variate using convolution method is:

1. Generate $Y_i \sim G(y)$, where G(y) is the distribution whose
2. Convolution of variates will give the desired distribution.
3. Sum the generated random variates, the sum will be the desired random variate.

$$X = \sum_{i=1}^{n} Y_i$$

4. Repeat from step 1 until you get the desired numbers of random variates.

6.6.4 Composition

Composition algorithm is useful for generating from compound distributions such as the hyper exponential or from compound Poisson processes. It is also frequently used to design specialized algorithms for generating from complicated densities. Composition method generates complex random variates by combining or transforming simpler random variates.

Assume that the pdf f(x) of X, from which samples are to be drawn, can be written (decomposed) in the form

$$f(x) = \sum_{i=1}^{r} p_i f_i(x)$$

Where

- the p_i form a discrete probability distribution, $\sum_i p_i = 1$
- the $f_i(x)$ are density functions, $\int f_i(x)\, dx = 1$

This kind of distribution is called a composition distribution. The sample generation can be done as follows

- draw index I from the distribution $\{p_1, p_2, \ldots, p_r\}$
- draw value of X using the pdf $f_I(x)$

The values of X are the random variates generated using this method.

1. The sequence of numbers 0.37, 0.29, 0.19, 0.88, 0.44, 0.63, 0.77, 0.70, 0.21 and 0.58 has been generated. Use K-S test to determine if the numbers are uniformly distributed. (The critical value for alpha=0.05 is 0.410 for sample size N=10) [2080 Shrawan]

Solution:

Given samples are: 0.37, 0.29, 0.19, 0.88, 0.44, 0.63, 0.77, 0.70, 0.21 are 0.58.

1. Setting up the hypothesis for uniformity as

   $H_0 = R_i \sim \cup [0, 1]$

   $H_1 = R_i \not\sim \cup [0, 1]$

2. Arranging the above random number in ascending order as:

   $0.19 < 0.2 < 0.29 < 0.37 < 0.44 < 0.58 < 0.63 < 0.70 < 0.77 < 0.88$

3. Computing D^+ and D^- from the following table:

(i)	1	2	3	4	5	6	7	8	9	10
R(i)	0.19	0.21	0.29	0.37	0.44	0.58	0.63	0.70	0.77	0.88
i/N	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$i/N - R_i$	–	–	0.01	0.03	0.06	0.02	0.07	0.1	0.13	0.12
$R_i - \frac{i-1}{N}$	0.19	0.11	0.09	0.07	0.04	0.08	0.03	0	–	–

$$D^+ = \max\left(\frac{i}{N} - R_i\right)$$

$$= \max\{0.01, 0.03, 0.06, 0.02, 0.07, 0.03, 0.13, 0.12\}$$

$$= 0.13$$

$$D^- = \text{Max}\left(R_i - \frac{i-1}{N}\right)$$

$$= \text{Max}\{0.19, 0.11, 0.07, 0.04, 0.08, 0.03, 0\}$$

$$= 0.19$$

4. $D = \text{Max}\{D^+, D^-\}$

$= \text{Max}\{0.13, 0.19\}$

$= 0.19$

5. Critical value of D at K = 0.05 and for N = 10 is D_α = 0.410

6. Since $D_\alpha > D$, H_0 is accepted.

2. Generate exponential variates X_i with mean 1 (λ=1), given random numbers R_i.

i	1	2	3	4	5
R_i	0.1306	0.0422	0.6597	0.7965	0.7696

Solution:

For exponential distribution the range (R_i) is given by

$$R = F(x_i) = 1 - e^{-\lambda X_i}$$

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

$$X_i = -\,ln(1 - R_i) \qquad \text{since } \lambda = 1$$

For simplification, we replace $1 - R_i$ by R_i to yield

$$X_i = -\,ln(R_i)$$

Using the values of R_1 provided in table, we get

$$X_1 = -\,ln(R_1)$$

$$= -\,ln(0.1306)$$

$$= 0.14$$

Doing same for all the values of i in R_i we get

X_i	0.1400	0.0431	1.078	1.592	1.468

3. **A sequence of 10,000 four-digit numbers has been generated and an analysis indicates the following combinations and frequencies: Use Poker's test to determine if these random numbers are independent, α=0.05 and degree of freedom=4 is 9.488.**

Combination Distribution (i)	Observed Frequency (O_i)
4 different digits	5120
3 like digits	75
4 like digits	15
1 pair	4230
2 pairs	560

[2080 Shrawan]

Solution:

Given, $\chi^2 = 9.488$

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ∪ independently

$H_1 = R_i$ ∦ independently

Since there are 4 digits, i.e., 4 Poker hands then,

Calculation of probability:

1. P (all same) = P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit same as 1^{st}) × P (4^{th} digit same as 1^{st})

   $= 1 \times 0.1 \times 0.1 \times 0.1$

   $= 0.001$

2. P (all 4-digit diff) = P (2^{nd} digit different from 1^{st}) × P (3^{rd} digit diff. from 1^{st} and 2^{nd}) × P (4^{th} digit diff as 1^{st}, 2^{nd} and 3^{rd})

   $= 1 \times 0.9 \times 0.8 \times 0.7$

   $= 0.504$

3. P (1 pair) = ${}^4C_2$ $(1 \times 0.9 \times 0.8 \times 0.1)$

   $= 0.432$

4. $P(2 \text{ pair}) = {}^4C_2/2 \times {}^2C_2 (1 \times 0.9 \times 0.1 \times 0.1)$
$= (4!/2!2!)/2 \times 1 (0.01 \times 0.9)$
$= 3 \times 0.09$ 0.027
$= 0.27$

5. $P(3 \text{ like digits}) = {}^4C_3 \times (1 \times 0.9 \times 0.1 \times 0.1)$
$= 4 \times (0.01 \times 0.9)$
$= 4 \times 0.09$
$= 0.036$

The expected value (E_i) for each combination is calculated by the following expression:

$E_i = \text{Probability} \times N$

With N = 10000, let's summarize the results of Poker's test in the following table:

Combination (i)	Observed frequency (O_i)	Expected frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
4 different digits	5120	5040	1.269
3 like digits	75	360	225.625
4 like digits	15	10	2.5
1 pair	4230	4320	1.875
2 pairs	560	270	311.48

From above distribution table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

$= 1.269 + 225.625 + 2.5 + 1.875 + 311.48$

$= 542.749$

At $\alpha = 0.05$ and degree of freedom = 4, the critical value of χ^2 is 9.488.

i.e. $\chi^2_{0.05,4} = 9.488$

Therefore, $\chi^2_{0.05,4} < \chi^2_0$, so H_0 is rejected.

Thus, the independence of the random numbers is rejected based on this test.

4. A set of 10,000 4-digit random values have been generated. An observation shows that 4731 values have all different digits, 2272 have 2 of a kind digit, 769 have 3 of a kind, 1547 have 2 pairs and 681 have all same digits. Test these values for randomness using the Poker test (use 9.49 as a critical value). [2079 Jestha]

Solution:

Given, $\chi^2 = 9.49$

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ∽ independently

$H_1 = R_i$ ⊀ independently

Since there are 4 digits, i.e., 4 Poker hands then,

Calculation of probability:

1. P (all same) = P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit same as 1^{st}) × P (4^{th} digit same as 1^{st})
   $= 1 \times 0.1 \times 0.1 \times 0.1$
   $= 0.001$

2. P (all 4-digit diff) = P (2^{nd} digit different from 1^{st}) × P (3^{rd} digit diff. from 1^{st} and 2^{nd}) × P (4^{th} digit diff as 1^{st}, 2^{nd} and 3^{rd})
   $= 1 \times 0.9 \times 0.8 \times 0.7$
   $= 0.504$

3. P (1 pair) = ${}^4C_2$ $(1 \times 0.9 \times 0.8 \times 0.1)$
   $= 0.432$

4. P (2 pair) = ${}^4C_2/2 \times 2C2$ $(1 \times 0.9 \times 0.1 \times 0.1)$
   $= (4!/2!2!)/2 \times 1\ (0.01 \times 0.9)$
   $= 3 \times 0.09 = 0.27$ 0.027

5. P (3 like digits) $= {}^4C_3 \times (1 \times 0.9 \times 0.1 \times 0.1)$

$= 4 \times (0.01 \times 0.9)$

$= 4 \times 0.09$

$= 0.036$

The expected value (E_i) for each combination is calculated by the following expression:

E_i = Probability × N

With N = 10000, let's summarize the results of Poker's test in the following table:

Combinations (i)	Observed Frequency (O_i)	Expected Frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
All different	4731	5040	18.94
2 of a kind	2272	4320	970.90
3 of a kind	769	360	464.66
2 pairs	1547	270	6039.73
All same	681	10	45024.1

From the above distribution table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

$= 18.94 + 970.90 + 464.66 + 6039.73 + 45024.1$

$= 52518.63$

From Given, $\chi^2 = 9.49$

Therefore, $X^2_{0.05,4} < \chi_0^2$, so H_0 is rejected.

Thus, the independence of the random numbers is rejected based on this test.

5. **A sequence of 1000 four-digit numbers were generated as follows:**

Combinations (i)	Observed Frequency
All different digits	540
One pair	320
Two pair	70
Three of a kind	50
All same digits	20

Use poker test to check independence. (Use tabulated value as 9.488 for confidence of 95% and N=4). [2078 Chaitra]

Solution:

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ∽ independently

$H_1 = R_i$ ⊀ independently

Since there are 4 digits, i.e., 4 Poker hands then,

Calculation of probability:

1. P (all same) = P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit same as 1^{st}) × P (4^{th} digit same as 1^{st})

$= 1 \times 0.1 \times 0.1 \times 0.1$

$= 0.001$

2. P (all 4-digit diff) = P (2^{nd} digit different from 1^{st}) × P (3^{rd} digit diff. from 1^{st} and 2^{nd}) × P (4^{th} digit diff as 1^{st}, 2^{nd} and 3^{rd})

$= 1 \times 0.9 \times 0.8 \times 0.7$

$= 0.504$

3. P (1 pair) $= {}^4C_2 (1 \times 0.9 \times 0.8 \times 0.1)$

$= 0.432$

4. P (2 pair) $= {}^4C_2/2 \times {}^2C_2 (1 \times 0.9 \times 0.1 \times 0.1)$
$= (4!/2!2!)/2 \times 1 (0.01 \times 0.9)$
$= 3 \times 0.09$
$= 0.27$ 0.027

5. P (3 like digits) $= {}^4C_3 \times (1 \times 0.9 \times 0.1 \times 0.1)$
$= 4 \times (0.01 \times 0.9)$
$= 4 \times 0.09$
$= 0.036$

The expected value (E_i) for each combination is calculated by the following expression:

E_i = Probability × N

With N = 1000, let's summarize the results for Poker's test in the following table:

Combinations (i)	Observed Frequency (O_i)	Expected Frequency (E_i)	$\frac{(O_i - E_i)}{E_i}$
All different digits	540	504	2.571
One pair	320	432	29.037
Two pair	70	27	68.48
Three of a kind	50	36	5.44
All same digits	20	1	361

From above distribution table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

$= 2.571 + 29.037 + 68.48 + 5.44 + 361$

$= 466.258$

For 95% of confidence and degree of freedom=4, the critical value of χ^2 is 9.488

i.e., $\chi^2_{0.05,4} = 9.488$

Therefore, $\chi^2_{0.05,4} < \chi^2_0$, so H_0 is rejected.

Thus, the independence of the random numbers is rejected based on this test.

6. A sequence of 1000 four-digit numbers were generated as follows:

Combinations (i)	Observed Frequency
4 different digits	540
3 like digits	50
4 like digits	20
1 pair	320
2 pairs	70

Use poker test to determine if these random numbers are independent at $\alpha = 0.05$.

Given critical value of χ^2 at $\alpha = 0.05$ and degree of freedom = 4 is 9.49. [2077 Chaitra, 2075 Bhadra]

Solution:

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ∽ independently

$H_1 = R_i$ ⊁ independently

Since there are 4 digits, i.e., 4 Poker hands then,

Calculation of probability:

1. P (all same) = P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit same as 1^{st}) × P (4^{th} digit same as 1^{st})

$= 1 \times 0.1 \times 0.1 \times 0.1$

$= 0.001$

2. P (all 4-digit diff) = P (2^{nd} digit different from 1^{st}) × P (3^{rd} digit diff. from 1^{st} and 2^{nd}) × P (4^{th} digit diff as 1^{st}, 2^{nd} and 3^{rd})

$= 1 \times 0.9 \times 0.8 \times 0.7$

$= 0.504$

3. P (1 pair) $= {}^4C_2\ (1 \times 0.9 \times 0.8 \times 0.1)$

$= 0.432$

4. P (2 pair) $= {}^4C_2/2 \times 2C2\ (1 \times 0.9 \times 0.1 \times 0.1)$

$= (4!\ /2!2!)/2 \times 1\ (0.01 \times 0.9)$

$= 3 \times 0.09$ 0.027

$= 0.27$

5. P (3 like digits) $= {}^4C_3 \times (1 \times 0.9 \times 0.1 \times 0.1)$

$= 4 \times (0.01 \times 0.9)$

$= 4 \times 0.09$

$= 0.036$

The expected value (E_i) for each combination is calculated by the following expression:

E_i = Probability × N

With N=1000, let's summarize the results for Poker's test in the following table:

Combinations (i)	Observed Frequency (O_i)	Expected Frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
4 different digits	540	504	2.571
3 like digits	50	36	5.44
4 like digits	20	1	361
1 pair	320	432	29.037
2 pairs	70	27	68.48

From above distribution table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

$= 2.571 + 5.44 + 361 + 29.037 + 68.48$

$= 466.258$

At $\alpha = 0.05$ and degree of freedom=4, the critical value of χ^2 is 9.49

i.e., $\chi^2_{0.05,4} = 9.49$

Therefore, $\chi^2_{0.05,4} < \chi^2_0$, so H_0 is rejected.

Thus, the independence of the random numbers is rejected based on this test.

---

7. Use the Kolmogorov-Smirnov test for the following random numbers with level of significance of $\alpha = 0.05$ is 0.432 for sample size N = 9. Random numbers are. 0.37, 0.55, 0.71, 0.97, 0.65, 0.29, 0.84, 0.78, 0.23 [2077 Chaitra]

Solution:

Given samples are:

0.37, 0.55, 0.71, 0.97, 0.65, 0.29, 0.84, 0.78, 0.23

1. Setting up the hypothesis for uniformity as

   $H_0 = R_i \sim U\ [0, 1]$

   $H_1 = R_i \not\sim U\ [0, 1]$

2. Arranging the above random numbers in ascending order as:

   $0.23 < 0.29 < 0.37 < 0.55 < 0.65 < 0.71 < 0.78 < 0.84 < 0.97$

3. Computing D^+ and D^- from the following table:

(i)	1	2	3	4	5	6	7	8	9
R_i	0.23	0.29	0.37	0.55	0.65	0.71	0.78	0.84	0.97
i/N	0.11	0.22	0.33	0.44	0.55	0.67	0.78	0.89	1
$i/N - R_i$	-	-	-	-	-	-	0	0.05	0.03
$R_i - \frac{i-1}{N}$	0.23	0.17	0.14	0.21	0.20	0.15	0.11	0.06	0.08

$D^+ = \max \{i/N - R_i\}$

$= \max \{0, 0.05, 0.03\}$

$= 0.05$

$D^- = \max \left\{R_i - \frac{i-1}{N}\right\}$

$= \max \{0.23, 0.17, 0.14, 0.21, 0.20, 0.15, 0.11, 0.06, 0.08\}$

$= 0.23$

4. $D = \max\{D^+, D^-\}$

$= \max \{0.05, 0.23\}$

$= 0.23$

5. Critical value of D at $\alpha = 0.05$ and for $N = 9$ is

$D_\alpha = 0.432$

6. Since $D_\alpha > D$, so H_0 is accepted.

Thus, the distribution is uniform.

A sequence of 10,000 four-digit numbers has been generated and an analysis indicates the following combinations and frequencies: Use Poker's test to determine if these random numbers are independent, α=0.05 and degree of freedom=4 is 7.78.

Combinations (i)	Observed Frequency
4 different digits	5000
3 like digits	700
4 like digits	500
1 pair	3000
2 pairs	800

[2076 Bhadra]

Solution:

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ~ independently

$H_1 = R_i$ ≁ independently

Since there are 4 digits, i.e., 4 Poker hands then,

Calculation of probability:

1. P (all same) = P (2^{nd} digit same as 1^{st}) × P (3^{rd} digit same as 1^{st}) × P (4^{th} digit same as 1^{st})

$= 1 \times 0.1 \times 0.1 \times 0.1$

$= 0.001$

2. P (all 4-digit diff) = P (2^{nd} digit different from 1^{st}) × P (3^{rd} digit diff. from 1^{st} and 2^{nd}) × P (4^{th} digit diff as 1^{st}, 2^{nd} and 3^{rd})

$= 1 \times 0.9 \times 0.8 \times 0.7$

$= 0.504$

3. P (1 pair) $= {}^4C_2 (1 \times 0.9 \times 0.8 \times 0.1)$

$= 0.432$

4. P (2 pair) $= {}^4C_2/2 \times {}^2C_2 (1 \times 0.9 \times 0.1 \times 0.1)$

$= (4!/2!2!)/2 \times 1 (0.01 \times 0.9)$

$= 3 \times 0.09$ 0.027

$= 0.27$

5. P (3 like digits) $= {}^4C_3 \times (1 \times 0.9 \times 0.1 \times 0.1)$

$= 4 \times (0.01 \times 0.9)$

$= 4 \times 0.09$

$= 0.036$

The expected value (E_i) for each combination is calculated by the following expression:

E_i = Probability × N

With N = 10000, let's summarize the results for Poker's test in the following table:

Combinations (i)	Observed Frequency (O_i)	Expected Frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
4 different digits	5000	5040	0.317
3 like digits	700	360	321.11
4 like digits	500	10	24010
1 pair	3000	4320	403.33
2 pairs	800	270	1040.37

From above distribution table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

$$= 0.317 + 321.11 + 24010 + 403.33 + 1040.37$$

$$= 25775.12$$

At $\alpha = 0.05$ and degree of freedom=4, the critical value of χ^2 is 7.78

i.e., $\chi^2_{0.05,4} = 7.78$

Therefore, $\chi^2_{0.05,4} < \chi_0^2$, so H_0 is rejected.

Thus, the independence of the random numbers is rejected based on this test.

9. The generated random numbers are 0.206, 0.389, 0.915, 0.287, 0.443, 0.767, 0.109, 0.177, 0.999, 0.0156. Use Kolmogorov- Smirnov test to check whether the numbers are uniformly distributed or not at a confidence level of 95%. (Note the critical value of D for α=0.05 and N=10 is 0.410)

[2076 Bhadra]

Solution:

Given samples are: 0.206, 0.389, 0.915, 0.287, 0.443, 0.767, 0.109, 0.177, 0.999, 0.0156

1. Setting up the hypothesis for uniformity as

$H_0 = R_i \sim U[0, 1]$

$H_1 = R_i \not\sim U[0, 1]$

2. Arranging the above random numbers in ascending order as:

$0.0156 < 0.109 < 0.177 < 0.206 < 0.287 < 0.389 < 0.443 < 0.767 < 0.915 < 0.999$

3. Computing D^+ and D^- from the following table:

(i)	1	2	3	4	5	6	7	8	9	10
R_i	0.0156	0.109	0.177	0.206	0.287	0.389	0.443	0.767	0.915	0.999
i/N	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$i/N - R_i$	0.084	0.091	0.123	0.194	0.213	0.211	0.257	0.033	-	0.001
$R_i - \frac{i-1}{N}$	0.0156	0.009	-	-	-	-	-	0.067	0.115	0.099

$D^+ = \max \{i/N - R_i\}$

$= \max \{0.084, 0.091, 0.123, 0.194, 0.213, 0.211, 0.257, 0.033, 0.001\}$

$= 0.257$

$D^- = \max \left\{R_i - \frac{i-1}{N}\right\}$

$= \max \{0.0156, 0.009, 0.067, 0.115, 0.099\}$

$= 0.115$

4. $D = \max \{D^+, D^-\}$

$= \max \{0.257, 0.115\}$

$= 0.257$

5. Critical Value of D at α=0.05 and for N=10 is

$D_\alpha = 0.410$

Since $D_\alpha > D$, so H_0 is accepted

Thus, the distribution is uniform.

10. Using Chi-square test the uniformity at 90% for the given random numbers. Degree of freedom for 6 = 10.645, 7 = 12.017, 8 = 13.362, 9 = 14.684, 10 = 15.987.

20	34	43	42	14	10	33	17	6	11
15	16	4	1	35	22	9	46	37	57
51	49	40	27	59	5	44	19	41	55
53	29	3	31	48	8	56	28	12	7

[2074 Bhadra]

Solution:

Here total observations (N) = 40

Let us divide the observations into 8 equal class intervals as shown below:

Class interval	Observed frequency (O_i)	Expected frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
1 – 8	7	5	0.8
9 – 16	7	5	0.8
17 – 24	4	5	0.2
25 – 32	4	5	0.2
33 – 40	5	5	0
41 – 48	6	5	0.2
49 – 56	5	5	0
57 – 64	2	5	1.8
	$\Sigma O_i = 40$	$\Sigma E_i = 40$	$\sum \frac{(O_i - E_i)^2}{E_i} = 4$

Here, the calculated value $(\chi_0^2) = \sum \frac{(O_i - E_i)^2}{E_i} = 4$

Critical value of χ^2 at $n-1 = 7$ degrees of freedom

$= \chi^2_{\alpha, n-1} = 12.017$ (given)

Since, $\chi^2_{\alpha, n-1} > \chi^2_0$, the test for uniformity is accepted.

11. **A sequence of 10,000 five-digit numbers has been generated and analysis indicates the following combinations and frequencies. Based on Poker test, check whether the number are independent. Use χ^2=12.592 (Use $\alpha = 0.05$ and N = 6 is 12.592).**

Combinations (i)	Observed Frequency
All different	3044
One pair	5020
Two pair	1090
Three of a kind	700
Full House	95
Four of a kind	40
Five of a kind	11
Total	10,000

[2074 Magh, 2072 Ashwin]

Solution:

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ∽ independently

$H_1 = R_i$ ⅄ independently

Since there are 5 digits, i.e., 5 Poker hands then,

Calculation of probability:

1. P (All different) = $1 \times 0.9 \times 0.8 \times 0.7 \times 0.6 = 0.3024$
2. P (1 Pair / 3 diff digits) = ${}^5C_2 (1 \times 0.9 \times 0.8 \times 0.7 \times 0.1)$
   $= {}^5C_2 (0.09 \times 0.56)$
   $= 0.5040$
3. P (2 pairs) = ${}^5C_2 /2 \times {}^3C_2 (1 \times 0.9 \times 0.8 \times 0.1 \times 0.1)$
   $= (5! /2!3!)/2 \times 3 \times (0.01 \times 0.72)$
   $= 10/2 \times 3(0.0072)$
   $= 0.108$
4. P (three of a kind) = ${}^5C_3 (1 \times 0.9 \times 0.8 \times 0.1 \times 0.1)$
   $= 5! /2!3! \times (0.01 \times 0.72)$
   $= 0.072$
5. P (Full House) = ${}^5C_3 \times {}^2C_2 (1 \times 0.9 \times 0.1 \times 0.1 \times 0.1)$
   $= 10 \times 1 (0.001 \times 0.9)$
   $= 0.009$
6. P (Four of a kind) = ${}^5C_4 (1 \times 0.9 \times 0.1 \times 0.1 \times 0.1)$
   $= 5 (0.001 \times 0.9)$
   $= 0.045$ 0.0005
7. P (five of a kind/all same) = $1 \times 0.1 \times 0.1 \times 0.1 \times 0.1$
   $= 0.0001$

The expected value (E_i) for each combination is calculated by the following expression:

E_i = Probability $\times$ N

With N=10000, let's summarize the results for Poker's test in the following table:

Combinations (i)	Observed Frequency (O_i)	Expected Frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
All different	3044	3024	0.1322
One pair	5020	5040	0.079
Two pair	1090	1080	0.092
Three of a kind	700	720	0.556
Full House	95	90	1040.37
Four of a kind	40	45	0.556
Five of a kind	11	1	100

From above distribution table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

$$= 0.1322+0.079+0.092 + 0.556 + 1040.37 + 0.556 + 100$$

$$= 1141.78$$

At $\alpha = 0.05$ and degree of freedom=6, the critical value of χ^2 is 12.592

i.e., $\chi^2_{0.05,6} = 12.592$

Therefore, $\chi^2_{0.05,6} < \chi^2_0$, so H_0 is rejected.

Thus, the independence of the random numbers is rejected based on this test.

12. The sequence of numbers 0.63, 0.49, 0.24, 0.57, 0.71, 0.89 has been generated. Use the Kolmogorov-Smirnov test with α=0.05 to determine, if the hypothesis that the numbers are evenly distributed on interval [0, 1] can be rejected. (Given critical value of D for α=0.05 & N=6 is 0.521)

Solution:

Given samples are: 0.63, 0.49, 0.24, 0.57, 0.71, 0.89

1. Setting up the hypothesis for uniformity as

   $H_0 = R_i \sim U[0, 1]$

   $H_1 = R_i \not\sim U[0, 1]$

2. Arranging the above random numbers in ascending order as:

   $0.24 < 0.49 < 0.57 < 0.63 < 0.71 < 0.89$

3. Computing D^+ and D^- from the following table:

(i)	1	2	3	4	5	6
R_i	0.24	0.49	0.59	0.63	0.71	0.89
i/N	0.17	0.33	0.5	0.67	0.83	1
$i/N - R_i$	-	-	-	0.04	0.12	0.11
$R_i - \frac{i-1}{N}$	0.24	0.32	0.24	0.13	0.04	0.06

$D^+ = \max \{i/N - R_i\}$

$= \max \{0.04, 0.12, 0.11\}$

$= 0.12$

$D^- = \max \{R_i - \frac{i-1}{N}\}$

$= \max \{0.24, 0.32, 0.24, 0.13, 0.04, 0.06\}$

$= 0.32$

4. $D = \max \{D^+, D^-\}$

   $= \max \{0.12, 0.32\}$

   $= 0.32$

5. Critical value of D at $\alpha=0.05$ and for N=6 is

   $D_\alpha = 0.521$ (from K-S table)

6. Since $D_\alpha > D$ so H_0 is accepted

   Thus, the distribution is uniform.

13. **A set of 100 random numbers have been generated as below:**

0.34	0.90	0.25	0.89	0.87	0.44	0.12	0.21	0.46	0.67
0.83	0.76	0.79	0.64	0.7	0.81	0.94	0.74	0.22	0.74
0.96	0.99	0.77	0.67	0.56	0.41	0.52	0.73	0.99	0.02
0.47	0.3	0.17	0.82	0.56	0.05	0.45	0.31	0.78	0.05
0.79	0.71	0.23	0.19	0.82	0.93	0.65	0.37	0.39	0.42
0.99	0.17	0.99	0.46	0.02	0.66	0.1	0.42	0.18	0.49
0.37	0.51	0.54	0.01	0.81	0.25	0.69	0.34	0.75	0.49
0.72	0.43	0.56	0.97	0.3	0.94	0.96	0.58	0.73	0.05
0.06	0.39	0.84	0.24	0.4	0.64	0.4	0.19	0.79	0.62
0.18	0.26	0.97	0.88	0.64	0.47	0.6	0.11	0.29	0.78

Use Chi-Square test to test whether the data shown above are uniformly distributed or not. (Use α=0.05)

Solution:

Total samples (observations) = N = 100

Let's divide the above random number into a total of 10 intervals i.e. [0 – 0.1], [0.1 – 0.2], …, [0.9, 1]

Setting up the hypothesis for uniformity as

$H_0 = R_i \backsim U\,[0, 1]$

$H_1 = R_i \not\backsim U\,[0, 1]$

The expected frequency (E_i) is calculated by

$E_i = N/n = 100/10 = 100$

n is chosen in such a way that n >= 5

Dividing the above random numbers into 10 intervals from 0 to 1 as in the following table with respective observed and expected frequencies as

S. N	Interval	Observed Frequency (O_i)	Expected Frequency (E_i)	$\frac{(O_i - E_i)^2}{E_i}$
1	0-0.1	8	10	0.4
2	0.1-0.2	8	10	0.4
3	0.2-0.3	10	10	0
4	0.3-0.4	9	10	0.1
5	0.4-0.5	12	10	0.4
6	0.5-0.6	8	10	0.4
7	0.6-0.7	10	10	0
8	0.7-0.8	14	10	1.6
9	0.8-0.9	10	10	0
10	0.9-1	11	10	0.1

From the above table,

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} = 3.4$$

At $\alpha = 0.05$ and degree of freedom=9, the critical value of χ^2 is 16.9

i.e., $\chi^2_{0.05,9} = 16.9$

Therefore, $\chi^2_{0.05,9} > \chi_0^2$ so H_0 is accepted

Thus, the distribution is uniform.

14. Consider the following sequence of 120 digits:
2, 3, 6, 5, 6, 0, 0, 1, 3, 4, 5, 6, 7, 9, 4, 9, 3, 1, 8, 3, 1, 3, 7, 4, 8, 6, 2, 5, 1, 6, 4, 4, 3, 3, 4, 2, 1, 5, 8, 7, 0, 8, 8, 2, 6, 7, 8, 1, 3, 5, 3, 8, 4, 0, 9, 0, 3, 0, 9, 2, 4, 6, 9, 9, 8, 5, 6, 0, 1, 7, 6, 7, 0, 3, 1, 0, 2, 4, 2, 0, 1, 1, 2, 6, 7, 6, 3, 7, 5, 9, 3, 6, 6, 7, 8, 2, 3, 5, 9, 6, 6, 4, 0, 3, 9, 3, 6, 8, 1, 5, 0, 7, 6, 2, 6, 0, 5, 7, 8, 0
Test whether these digits would be assumed to be independent based on frequency with which gap occurs. (Use α=0.05)

Solution:

Setting up the hypothesis for testing independence as

$H_0 = R_i$ ∽ independently

$H_1 = R_i$ ∽̸ independently

Total no of gaps (N) = No of digits (total) – No of distinct digits

$= 120 - 10$

$= 110$

Determination of gap length:

Digits	Length of Each Gap	No of Gaps
0	0, 33, 12, 1, 1, 9, 4, 2, 3, 22, 7, 3, 4	13
1	9, 2, 7, 7, 10, 20, 5, 5, 0, 26	10
2	25, 8, 7, 15, 1, 3, 12, 17	9
3	6, 7, 2, 1, 10, 0, 14, 1, 5, 16, 12, 3, 5, 6, 1	15
4	4, 8, 6, 0, 2, 17, 7, 16, 23	9
5	6, 16, 9, 11, 15, 22, 8, 11, 6	9
6	1, 6, 13, 3, 14, 16, 4, 3, 12, 1, 5, 0, 6, 0, 5, 5, 1	17
7	9, 16, 5, 23, 1, 12, 2, 5, 17, 5	10
8	5, 13, 2, 0, 3, 4, 12, 29, 12, 10	10
9	1, 38, 3, 3, 0, 25, 8, 5	8

The relative frequency is calculated as: $\frac{\text{Frequency}}{N}$

The cumulative relative frequency ($S_N(x)$) is calculated as:

$\frac{\text{No of gaps} \le x}{\text{Total no of gaps}}$, $x = 3, 7, 11, 15, 19, 23, 27, 31, 35, 39$

The theoretical frequency (F (x)) is calculated as:

$1 - 0.9^{x+1}$

Cumulative Distribution Table (using above table):

Gap Length	Frequency	Relative Frequency	Cumulative Relative Frequency (S_N (x))	Theoretical frequency (F (x))	$\|F(x) - S_N(x)\|$
0-3	34	0.309	0.309	0.3439	0.0349
4-7	30	0.27	0.58	0.5695	0.01
8-11	13	0.1181	0.7	0.7175	0.0175
12-15	13	0.1181	0.8146	0.8146	$3.4' 10^{-3}$
16-19	9	0.0818	0.9	0.8746	0.0216
20-23	5	0.045	0.945	0.92	0.025
24-27	3	0.027	0.972	0.947	0.025
28-31	1	9.09×10^{-3}	0.981	0.9656	0.015
32-35	1	9.09×10^{-3}	0.99	0.0977	0.013
36-39	1	9.09×10^{-3}	1.00	0.9852	0.0148

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

$= \max \{0.0349, 0.01, 0.0175, 3.4\times10^{-3}, 0.0216, 0.025, 0.025, 0.015, 0.013, 0.0148\}$

$= 0.0349$

At 5% LOS with N=110, critical value of D is 0.129

i.e., $D_{0.05} = \frac{1.36}{\sqrt{N}} = \frac{1.36}{\sqrt{110}} = 0.129$

Therefore, $D_{0.05} > D$ so H_0 is accepted.

Thus, the random numbers are independently distributed.

Exercise

1. What are the basic properties of random numbers? [2080 Shrawan]
2. Write an algorithm of Kolmogorov-Smirnov test. [2079 Chaitra]
3. What do you mean by uniformity test? Explain the process of uniformity test of random numbers by the K-S method, using your example. [2078 Chaitra]
4. What are the different random number generation methods? Explain with examples. [2075 Bhadra, 2073 Bhadra]
5. What are the properties of random numbers? Explain the techniques of the generation of random numbers with appropriate examples. [2074 Bhadra]
6. Explain the different congruential random number generation methods with examples. Explain the rejection method. [2074 Magh]
7. Write an algorithm for Gap Test. [2074 Magh, 2072 Ashwin]
8. What do you mean by pseudo-random numbers? Explain the Gap test algorithm with an example. [2073 Magh]
9. What assumptions should we make while generating pseudo-random numbers? [2072 Magh]
10. Formulate a 4-digit poker test with suitable data with examples. [2071 Bhadra]
11. Why is the gap test used? [2071 Magh]
12. What are the problems associated with generating pseudo-random numbers? [2070 Bhadra]
13. Write a short note on convolution in random numbers. [2070 Bhadra]

CHAPTER 7

VERIFICATION AND VALIDATION OF SIMULATION MODELS

CHAPTER - 7

VERIFICATION AND VALIDATION OF SIMULATION MODELS

7.1 Verification and Validation

Verification and *validation* of a simulation model are crucial and challenging tasks for developers. Engineers, analysts, and managers who rely on the model outputs to make design recommendations or decisions often approach the model with skepticism regarding its validity. Therefore, it falls upon the model developer to work closely with end users during development and validation to address this skepticism and enhance the model's credibility.

The validation process has two primary objectives. First, it aims to create a model that closely represents the actual system's behavior, allowing it to be effectively used as a substitute for the real system. This enables experimentation, analysis of system behavior, and prediction of system performance. Second, the validation process strives to elevate the model's credibility to an acceptable level, ensuring that managers and decision makers are willing to rely on it.

Throughout the development and validation period, the model developer collaborates with end users to build trust and confidence in the model. By addressing user concerns, incorporating feedback, and demonstrating the model's accuracy, the developer can reduce skepticism and enhance the model's credibility. Ultimately, the successful verification and validation of a simulation model contribute to its acceptance and utilization by decision makers in various domains.

7.1.1 Verification

Verification of the model is the process of confirming that it is correctly implemented concerning the conceptual model i.e.,

whether the model matches the specifications and assumptions deemed acceptable for the given purpose of application.

The main objective of model verification is to ensure that the implementation of the model is correct.

Verification asks the following question:

- Is the conceptual model accurately represented by the operational model?
- Is the model implemented correctly in simulation software?
- Are the input parameters and logical structure of the model represented correctly?
- Are we building the model, right?

  Hence, verification is concerned with building the model correctly.

Example: Imagine you have built a simulation model of a traffic intersection, where you have included variables for the number of cars, the speed of the cars, and the traffic signals at the intersection. To verify your simulation model, you might test it under a variety of different conditions, such as different numbers of cars, different speeds, and different traffic signal timings. You can then compare the results of your simulation to what you would expect to see in the real world, based on your understanding of how traffic intersections work.

7.1.2 Validation

Validation refers to the process of assessing the accuracy and reliability of a model by comparing its output to real-world data or observations. The goal of validation is to determine the extent to which the model accurately represents the behavior, dynamics, and performance of the actual system being simulated. Validation asks the following question:

- Is the model an accurate representation of the real system?
- Are we building the right model?

  Hence, validation is concerned with building the correct model.

Example: Validating a simulation model of a traffic intersection would involve comparing the model's predictions of traffic flow and congestion to actual data collected from the real-world intersection. This could be done by comparing the model's predictions to measurements of traffic volumes, speeds, and delays taken at the intersection over time, or by comparing the model's predictions to observations of traffic patterns and behavior made by an observer at the intersection.

Validation is usually achieved through the calibration of the model, which is an iterative process of comparing the model to actual system behavior and using the discrepancies between the two, and the insights gained to improve the model. This process is repeated until model accuracy is acceptable.

Differences between Verification and Validation:

Verification	Validation
It is the process of determining a model implementation and its associated data accurately represent the developer's conceptual descriptions and specifications.	It is the process of determining the degree to which a model and its associated data are an accurate representation of the real world from the perspective of intended model uses.
Verification answers the question: "have we built the model, right?"	Validation answers the question: "Have we built the right model?"
Verification is a software engineering approach that checks the specifications and requirements of customers.	Validation is concerned with the accuracy of the model and checks the functionality of the simulation model in the real system.
Verification is the process of checking whether the software meets the specifications	Validation is the process of checking whether specifications capture the customer's needs

Verification	Validation
Verification includes all activities associated with high-quality software testing inspection, design analysis, specification analysis, and so on.	Validation involves making subjective assessments of how well the system (purposed) addresses the real - world. It includes requirement modeling, prototyping, and user evaluation.
For example, during the verification of a simulation model, it is checked that the model is based on the correct mathematical equations, that the input and output data are correctly formatted, and that the simulation runs without errors.	For example, during the validation of a simulation model of a weather system, the model's output would be compared with actual weather data to see if it accurately predicts the weather patterns.

7.2 Verification of Simulation Models

Simulation models are often used to represent systems or processes to study their behavior, predict their performance, or optimize their design. The accuracy and reliability of simulation results depend on the validity of the model and the appropriateness of the simulation method. Therefore, it is important to verify that the simulation model is an accurate representation of the system or process being modeled.

Approaches to verifying simulation models:

- **Visual inspection:** This involves examining the model structure, equations, and input data to ensure that they are correct and consistent.
- **Comparison with experimental data:** The model can be compared with experimental data from the system or process being modeled to assess the accuracy of the model.

- **Sensitivity analysis:** This involves changing the input parameters of the model and observing the effect on the output, to determine the robustness and sensitivity of the model.
- **Testing:** The model can be tested using a set of test cases to ensure that it produces reasonable results and behaves as expected.
- **Peer review:** The model can be reviewed by other experts in the field to ensure that it is reasonable and well-founded.

Considerations to be used in Verification Process:

- The operational model should be checked by someone other than the developer, preferably an expert in the simulation software being used.
- Make a flow diagram that contains all logically possible actions a system can perform when an event occurs.
- Closely examine the model output for reasonableness under a variety of input parameters.
- Have the operational model print input parameters when the simulation ends to check if they are not altered.
- Make the operational model as self-documenting as possible.
- Verify animated operational model imitates the actual system.
- Debugger should be used during simulation model building.
- Graphical interfaces are recommended.

Verification of simulation models is an important step in the model development process, as it helps to ensure that the model is a valid representation of the system or process being studied.

7.3 Calibration and Validation of Models

Calibration is the iterative process of comparing the model to the real system behavior i.e., making adjustments or changes to the model, comparing the revised model to reality, and so on.

This comparison of the model to reality is carried out by subjective and objective tests.

- A subjective test involves talking to people, who are knowledgeable about the system making models and forming the judgment.
- An objective test involves one or more statistical tests to compare some model output with the assumptions in the model.

Hence, it attempts to confirm that a model represents the real system accurately.

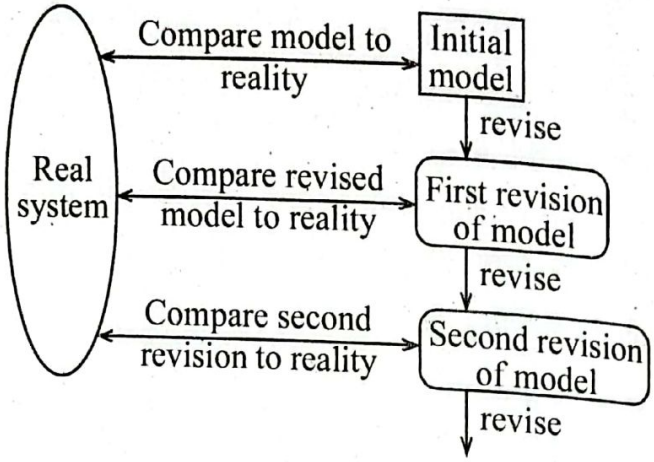


Fig.: Iterative process of calibrating a model

The iterative calibration of a model is important because it helps to improve the accuracy and reliability of the model. This process involves adjusting the model parameters until it provides results that are consistent with actual data. The goal of iterative calibration is to make the model more representative of the real-world system that it is modeling.

There are several reasons why iterative calibration is important:

- **Improving accuracy:**
  By adjusting the parameters of the model, the iterative calibration process helps to ensure that the model accurately represents the real-world system. This is critical for making reliable predictions about the behavior of the system.

- **Increasing realism:**
  Iterative calibration helps to refine the model so that it is more representative of the real-world system. This is important for ensuring that the model captures important characteristics and relationships in the system.

- **Reducing uncertainty:**
  The iterative calibration process can help to reduce the uncertainty associated with model predictions by improving the accuracy of the model. This is important for making informed decisions based on the results of the model.

Overall, the iterative calibration of a model is an important step in the modeling process because it helps to improve the reliability and accuracy of the model, which is critical for making informed decisions based on the results of the model.

7.4 Naylor and Finger Simulation Model Validation or Three-Step Validation

Naylor and Finger proposed a three-step approach for the validation process. These steps are as follows:

Step 1: Build a model that has high face validity.

Step 2: Validate model assumptions.

Step 3: Compare the model input-output transformations to corresponding input-output transformations for the real system.

1. Face Validity

The first goal of the simulation modeler is to construct a model that appears reasonable on its face to model users and others who are knowledgeable about the real system being simulated. The potential users of a model should be involved in model construction from its conceptualization to its implementation, to ensure that a high degree of realism is built into the model through reasonable assumptions

regarding system structure and through reliable data. Potential users and knowledgeable people can also evaluate model output for reasonableness and can aid in identifying model deficiencies. Thus, the users can be involved in the calibration process as the model is improved iteratively by the insights gained from identification of the initial model deficiencies. Another advantage of user involvement is the increase in the model's perceived validity, or credibility, without which a manager would not be willing to trust simulation results as a basis for decision making.

Sensitivity analysis can also be used to check a model's face validity. The model user is asked whether the model behaves in the expected way when one or more input variables is changed. For example, in most queueing systems, if the arrival rate of customers (or demands for service) were to increase, it would be expected that utilizations of servers, lengths of lines, and delays would tend to increase (although by how much might well be unknown). From experience and from observations on the real system (or similar related systems), the model user and model builder would probably have some notion at least of the direction of change in model output when an input variable is increased or decreased. For most large-scale simulation models, there are many input variables and thus many possible sensitivity tests. The model builder must attempt to choose the most critical input variables for testing if it is too expensive or time consuming to vary all input variables. If real system data are available for at least two settings of the input parameters, objective scientific sensitivity tests can be conducted via appropriate statistical techniques.

2. Validate Model Assumptions

Assumptions made about the model generally fall into two categories: *Structural assumptions* and *data assumptions.*

Structural assumptions:

Structural assumptions involve simplification and abstraction of reality. Assumptions made about how the system operates, how it is physically arranged, and what kind of model should be used are structural assumptions.

Data assumptions:

Data assumptions should be done based on the collection of reliable data. Assumptions made on what kind of input data model is, and what are the parameter values of the input data model.

Data should be verified to come from a reliable source. A typical error is assuming an inappropriate statistical distribution for the data. The assumed statistical model should be tested using the goodness of fit tests and other techniques.

Examples of the goodness of fit test are the K-S test and the Chi-Square test.

For example - Consider a bank system in which customers are queued before providing service.

The structural assumptions may be - Customers form a single queue which is served by multiple tellers. Customers form one line for each teller. The number of tellers should be fixed or variable.

These structural assumptions should be validated by actual observation during appropriate periods and also by discussions with the managers and tellers.

The data assumptions may be - interarrival times of customers during peak hours, interarrival times of customers during the slack period, service time for personal accounts, and so on.

These data assumptions should be validated by consultation with bank managers. The validation is done by using goodness-of-fits tests such as the chi-square test or K-S tests.

3. Validating Input-Output Transformations

It compares the model input-output transformations to the corresponding input-output transformations for the real system.

The model is viewed as an input – output transformation for these tests. The validation test consists of comparing outputs from the system under consideration to model output for the same set of input conditions. Data recorded while observing the system must be available to perform this test. Steps to validating input-output transformations:

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

For example, if the system under consideration is fast food drive-through where the input to the model is customer arrival time and output measure performance is average customer time in line, then arrival time and time spent in line for the customer at the drive-through would be recorded. These data are used to validate the model by giving customer arrival time as input to the model and comparing the output (average customer time in line) with the real-world recorded data.

7.5 Model Building, Verification, and Validation (From Modeling to Simulation)

- **The first step** in model building consists of observing the real system and the interactions among its various components and of collecting data on their behavior. But observation alone seldom yields sufficient understanding of system behavior. People familiar with the system, or any

subsystem, should be questioned, to take advantage of their special knowledge. Operators, technicians, repair and maintenance personnel, engineers, supervisors, and managers understand certain aspects of the system that might be unfamiliar to others. As model development proceeds, new questions may arise, and the model developers will return to this step to learn more about system structure and behavior.

The second step in model building is the construction of a conceptual model a collection of assumptions about the components and the structure of the system, plus hypotheses about the values of model input parameters. As is illustrated by Figure below, conceptual validation is the comparison of the real system to the conceptual model.

The third step is the implementation of an operational model, usually by using simulation software, and incorporating the assumptions of the conceptual model into the worldview and concepts of the simulation software.

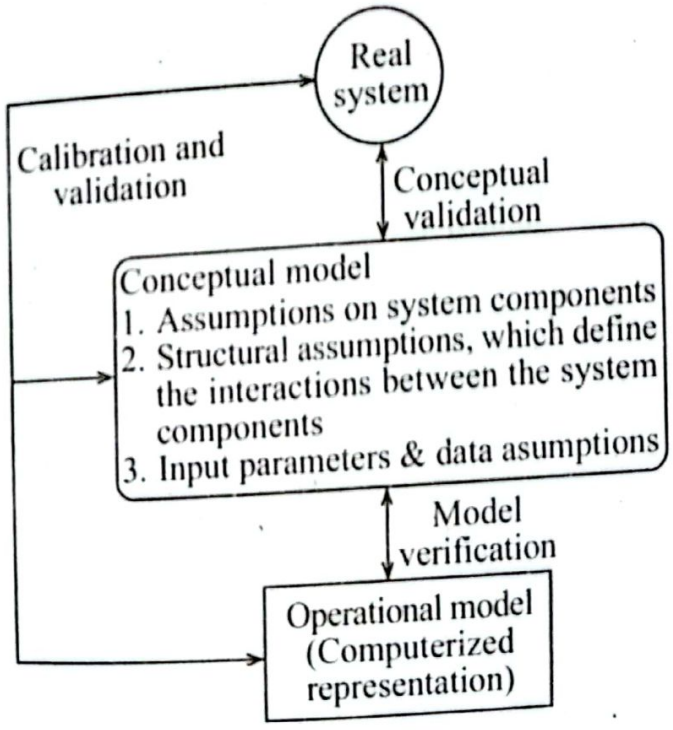


Fig.: Model building, verification, and validation[1]

In actuality, model building is not a linear process with three steps. Instead, the model builder will return to each of these steps many times while building, verifying, and validating the model. Figure above depicts the ongoing model building process, in which the need for verification and validation causes continual comparison of the real system to both the conceptual model and the operational model, and induces repeated modification of the model to improve its accuracy.

REFERENCES

[1] Jerry Banks et al. Discrete Event System Simulation. Fifth edition. Pearson Education Limited, 2014, p.312.

Exercise

1. Explain the validation of models in simulation. [2080 Shrawan]
2. Explain Naylor and Finger validation approach. [2079 Chaitra, 2073 Bhadra]
3. Explain three steps of validation in detail. [2079 Jestha]
4. Make comparison between verification and validation with example. [2078 Chaitra, 2076 Bhadra]
5. Explain the iterative process of calibrating a model. Why is it done? [2077 Chaitra, 2075 Bhadra, 2073 Magh]
6. Make comparisons between verification, validation, and calibration of a model using examples. [2074 Magh]
7. What are the calibration and validation of models? Explain with a practical example. [2074 Bhadra, 2072 Ashwin]
8. Explain steps in model building verification and validation with diagrams. [2072 Magh]

CHAPTER 8

ANALYSIS OF SIMULATION OUTPUT

CHAPTER -8

ANALYSIS OF SIMULATION OUTPUT

Simulation models are typically built using mathematical equations, logical rules, and assumptions that describe the behavior and interactions of the system's components. These models can range from simple, deterministic models to highly complex and dynamic models that incorporate uncertainty and randomness.

Randomness or stochasticity is often introduced into simulation models to account for the inherent variability and uncertainty present in real-world systems. Random variables, such as arrival times, service times, or failure rates, are used to represent uncertain elements of the system. By incorporating randomness, simulation models can generate a range of possible outcomes, allowing analysts to understand and analyze the system's behavior under different conditions.

Once randomness is introduced, the values of system variables in the simulation model can fluctuate as the simulation progresses. This fluctuation reflects the inherent uncertainty in the system and enables the exploration of different possible scenarios. To make sense of these fluctuating values, statistical analysis techniques such as descriptive statistics, hypothesis testing, and confidence intervals are used to analyze the simulation output and draw meaningful conclusions. In this chapter we will get insights on different simulation output analysis methods.

8.1 Estimation Methods

Statistical estimation methods are commonly used to estimate parameters from sample observations on random variables. Usually, a random variable is drawn from an infinite population that has stationary probability distribution with a mean, μ, and finite variance, σ^2. It means that population distribution is not affected by the number of samples being made, nor does it change with time. If the value of one sample is not affected by the value of another, the random variables are mutually independent.

The random variables which satisfy both of these conditions are said to be independently and identically distributed (i.i.d). The *central limit theorem* then could be applied for i.i.d variables for simulation data. The CLT states that "The sum of n i.i.d variables, drawn from a population that has a mean of μ and a variance of σ^2, is approximately distributed as a normal variable with a mean, $n\mu$, and a variance, $n\sigma^2$.

Any normal distribution can be transformed into a standard normal distribution, that has a mean of 0 and a variance of 1. Let x_i(i=1, 2, ..., n) be the n i.i.d. random variables. Using CLT and applying transformation gives the following approximate normal variate:

$$z = \frac{\sum_{i=1}^{n} x_i - n\mu}{\sigma\sqrt{n}}$$

In terms of sample mean $\bar{x}$

$$z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

Where,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Confidence interval estimation:

The variable $\bar{x}$ is the sample mean, which is the sum of the random variables so a confidence interval about its computed value needs to be established. The probability density function of the standard normal variate is shown in the figure below.

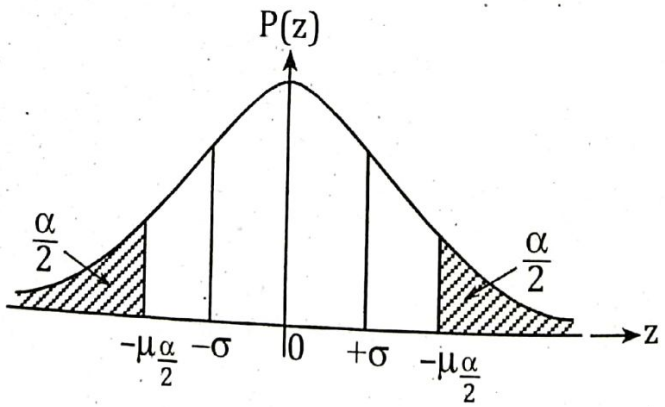


Fig.: Probability density function of standard normal variate

The integral from $-\infty$ to a value μ is the probability that z is less than or equal to μ, denoted by $P(\mu)$. Suppose $P(\mu) = 1 - \alpha/2$, where α is less than 1. Then the value of μ is denoted by $\mu_{\alpha/2}$. The probability that z is greater than $\mu_{\alpha/2}$ is then $\alpha/2$. The normal distribution is symmetric about its mean, so the probability that z is less than $-\mu_{\alpha/2}$ is also $\alpha/2$.

For some constant α, known as the level of significance, the probability that z lies between $-\mu_{\alpha/2}$ and $\mu_{\alpha/2}$ is given by

$$P\{-\mu_{\alpha/2} \leq z \leq \mu_{\alpha/2}\} = 1 - \alpha$$

Equivalently, in terms of the sample mean, we can write

$$P\left\{\bar{x} + \frac{\sigma}{\sqrt{n}}\mu_{\alpha/2} \geq \mu \geq \bar{x} - \frac{\sigma}{\sqrt{n}}\mu_{\alpha/2}\right\} = 1 - \alpha$$

The constant $1-\alpha$ is the confidence level and the confidence interval is

$$\bar{x} \pm \frac{\sigma}{\sqrt{n}}\mu_{\alpha/2}$$

In practice, the population variance σ^2 is usually not known; in that case, it is replaced by an estimate calculated as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

The normalized random variable based on σ^2 is replaced by a normalized random variable based on s^2. This has a Student-t distribution, with $n-1$ degrees of freedom.

The Student-t distribution is accurate only when the population from which the samples are drawn is normally distributed.

In terms of the estimated variance s^2, the confidence interval for $\bar{x}$ is

$$\bar{x} \pm \frac{s}{\sqrt{n}}t_{n-1,\,\alpha/2}$$

8.2 Simulation Run Statistics

In most of the simulation studies, the assumptions of stationary and mutually independent observations do not apply. An example of such a case is a queuing system where correlation is necessary to analyze the system. In such cases, the simulation run statistics method is used.

Consider a single-server system in which the arrivals occur with a Poisson distribution and the service time has an exponential distribution. Suppose the study objective is to measure the mean waiting time, defined as the time entities spend waiting to receive service and excluding the service time itself. This system is commonly denoted by M/M/1 which indicates:

first, that the inter-arrival time is distributed exponentially;
second, that the service time is distributed exponentially;
and third, there is one server.

The M stands for Markovian, which implies an exponential distribution.

In a simulation run, the simplest approach is to estimate the mean waiting time by accumulating the waiting time of n successive entities and dividing by n. This measure, the sample mean, is denoted by $\bar{x}(n)$ to emphasize the fact that its value depends upon the number of observations taken. If x_i (i = 1, 2, 3...., n) is the individual waiting times (including the value 0 for those entities that do not have to wait), then

$$\bar{x}(n) = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Whenever a waiting line forms, the waiting time of each entity on the line depends upon the waiting time of its predecessors. Any series of data that has this property of having one value affect other values is said to be *autocorrelated*. The sample mean of autocorrelated data can be shown to approximate a normal distribution as the sample size increases. The above equation for

$\bar{x}(n)$ remains a satisfactory estimate for the mean of autocorrelated data. A simulation run is started with the system in some initial state (mostly the idle state) in which no service is being given and no entities are waiting. The early arrivals then have a more than normal probability of obtaining service quickly, so a sample mean that includes the early arrivals will be biased. For a given sample size starting from a given initial condition, the sample mean distribution is stationary; but, if the distributions could be compared for different sample sizes, the distribution would be slightly different. The following figure is based on theoretical results, which show how the expected value of the sample mean depends upon the sample length, for the M/M/1 system, starting from an initially empty state, with a server utilization of 0.9.

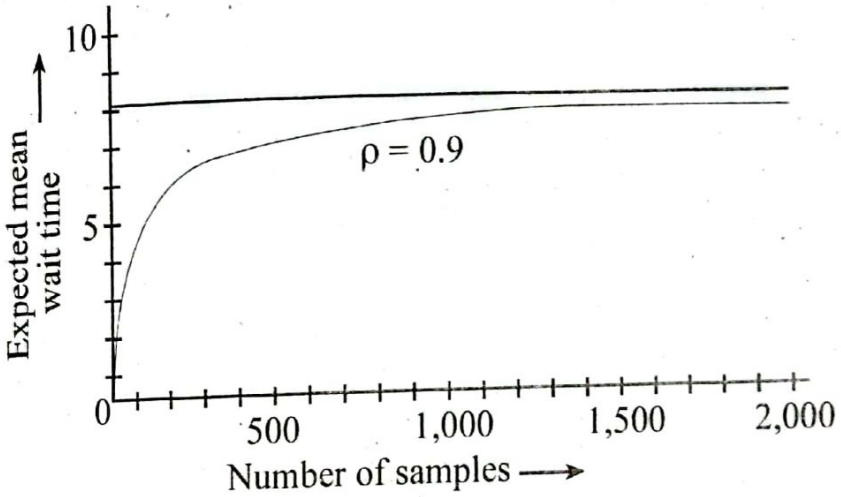


Fig.: Mean wait time in M/M/1 system for different sample sizes

Simulation run statistics help analysts understand the characteristics of the system being simulated, evaluate its performance, and make informed decisions based on the simulation output. They provide valuable insights into the behavior, variability, and distribution of the simulated system.

8.3 Replication of Runs

Replication of runs is used to obtain independent results by repeating the simulation. Repeating the experiment with different random numbers for the same sample size n gives a set of

independent determinations of the sample mean. The mean of the means and the mean of the variance are then used to estimate the confidence interval. The precision of the results of a dynamic stochastic can be increased by repeating the experiment with different random number. For each replication of a small sample size, the sample mean is determined. The sample means of the independent runs can be further used to estimate the variance of the distribution. Consider the experiment is repeated p times over the n random variables. Let x_{ij} be the i^{th} observation in j^{th} run, then the sample mean and variance for the j^{th} run are:

$$\bar{x}_j(n) = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

$$s_j^2 = \frac{1}{n-1} \sum_{i=1}^{n} [x_{ij} - \bar{x}_j(n)]^2$$

When we have similar means and variances for p independent measurements, then by combining them, the mean and variance for the population can be obtained as:

$$\bar{x} = \frac{1}{p} \sum_{j=1}^{p} \bar{x}_j(n)$$

$$s^2 = \frac{1}{p} \sum_{j=1}^{p} s^2{}_j(n)$$

The following figure shows the result of applying the procedure to experimental results for the M/M/1 system. Results are shown for server utilization 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6. In each case, the experiment was repeated from an initial idle state with different random numbers being used in each repetition. The result (figure below) shows the estimated mean waiting time as a function of sample size n.

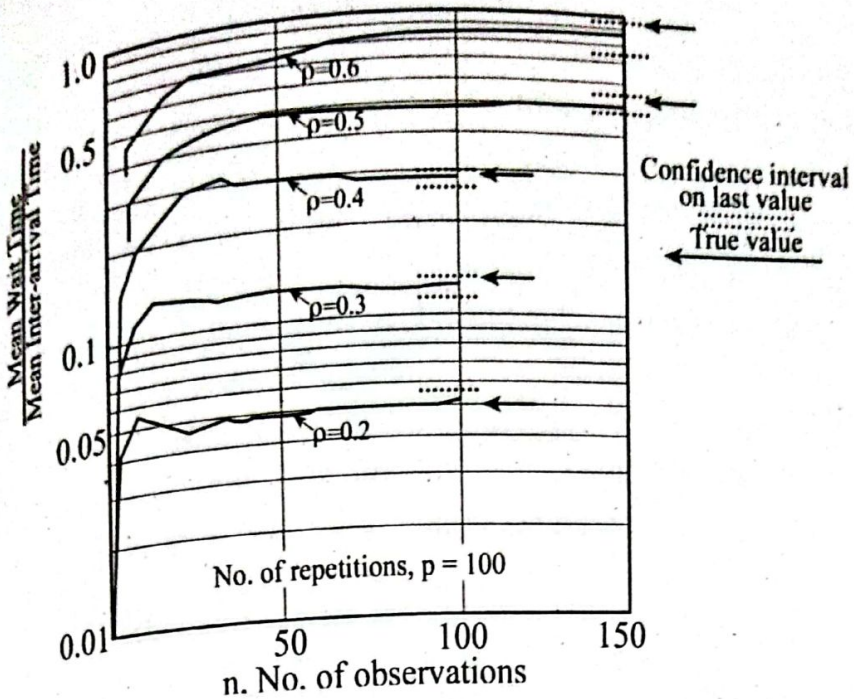


Fig.: Experimentally measured wait time in M/M/1 system for different sample sizes

The length of run of replications is so selected that all combined, it comes to the sample size N. i.e., p.n = N. By increasing the number of replications and shortening their length of run, the confidence interval can be narrowed. But due to the shortening of the length of replication, the effect of starting conditions will increase. The results obtained will not be accurate, especially when the initialization of the runs is not proper thus a compromise has to be made. There is no established procedure for dividing the sample size N into replications. However, it is suggested that the number of replications should not be very large and that the sample means should approximate a normal distribution.

8.4 Elimination of Initial Bias

Initial bias refers to the influence that the starting conditions of a simulation have on the results. Initial bias can occur when the starting conditions of the simulation are not representative of the population being studied, or when the starting conditions are not selected randomly.

When a simulation run is started in some initial conditions (State), frequently the idle state, in which no service is being given and no

entities are waiting, the early arrivals have more than normal probability of obtaining service quickly. So, the sample mean that includes early arrivals will be biased. Hence, the inclination of a higher probability of obtaining the service due to no entities waiting initially or idle in the queue is called initial bias. As the length of the simulation run is extended and the sample size increases, the biasing effect will decrease.

Elimination

Statistical biasing results from an unfair sampling of a population or from an estimation process that does not give accurate results on average. Two general approaches can be taken to remove the bias, the system can be started in a more representative state than the empty state, or the first part of the simulation can be ignored. The ideal situation is to know the steady state distribution for the system and select the initial condition from that distribution.

Some of the ways to eliminate the initial bias are as follows:

- **Ignore the initial data:** Ignore the initial bias that occurred during the simulation run.
- **Use representative state:** Start the system in a representative state than in an empty state.
- **Use randomization:** By randomly selecting the starting conditions of the simulation, it is less likely that initial bias will occur.
- **Use multiple starting points:** By running the simulation with multiple different starting points, it is possible to average out any initial bias that may be present.
- **Use a "burn-in" period:** A "burn-in" period is a period of time at the beginning of the simulation during which the results are not recorded. This allows the simulation to reach a more stable state before the results are collected.
- **Use a warm-up period:** A warm-up period is a period of time at the beginning of the simulation during which the

results are collected, but are not included in the final analysis. This allows the simulation to reach a more stable state before the results are used to estimate population parameters.

- **Use an equilibration period:** An equilibration period is a period of time at the beginning of the simulation during which the system is allowed to reach equilibrium before the results are collected. This can be useful when studying systems that are in a state of dynamic equilibrium.

Example

One example of eliminating initial bias in a simulation model could be in a financial model. If a financial model is being created to estimate the return on investment for a new project, the model may have certain initial assumptions about the costs and revenue of the project. However, these assumptions may not accurately reflect the actual costs and revenue of the project, leading to a bias in the model's prediction. To eliminate this initial bias, the model can be calibrated iteratively by comparing the model's predictions with actual data and adjusting the model's assumptions accordingly. This process can be repeated until the model's predictions are consistent with the actual data. By eliminating initial bias in this way, the model becomes more accurate and reliable, providing more valuable insights into the potential return on investment of the project.

Difference Between Estimation Method and Replication of Runs

Estimation Method	Replication of Runs
Estimation method involves the calculation of a single value that represents the quantity of interest, such as the mean, median, or standard deviation.	Replication of runs involves repeating the simulation several times to estimate the uncertainty of the results.

Estimation Method	Replication of Runs
This method is based on statistical techniques to estimate the parameters of a model using available data.	This method provides a more accurate estimate of the variance in the system, as it takes into account the random fluctuations in the model.
The accuracy of the estimation depends on the sample size and the quality of the data.	The number of replications required to achieve a reliable estimate depends on the complexity of the system and the desired level of confidence in the results.
The accuracy of the estimate is typically based on the quality and quantity of data used, as well as the sophistication of the estimation method.	Typically involves a large number of runs, often hundreds or thousands, to obtain a statistically meaningful representation of the variability in the results.
Examples are: Maximum likelihood estimation, Least squares estimation, Bayesian estimation.	Examples are: Monte Carlo simulation, Jackknife resampling, Cross-validation, Bootstrapping.

Exercise

1. How you can analyze simulation output using simulation runs statistics? Explain. [2080 Shrawan, 2074 Bhadra]
2. Explain the elimination of initial bias with an example. [2079 Chaitra, 2079 Jestha]
3. What are the various methods used to analyze simulation output? Explain any one of them. [2078 Chaitra]
4. Explain replications of runs in simulation output analysis. [2077 Chaitra]
5. Differentiate between the estimation method and the replication of runs. [2076 Bhadra]
6. How can you use estimation methods in the analysis of simulation output? Explain with an example. [2075 Bhadra, 2073 Magh]
7. Why do we need replications of runs? Explain with an example. [2074 Magh]
8. Explain the simulation run statistics with examples. [2073 Bhadra]
9. Define initial bias. Explain the methods for the elimination of initial bias. [2072 Ashwin]

CHAPTER 9

SIMULATION SOFTWARE

CHAPTER - 9

SIMULATION SOFTWARE

Simulation software is a computer program or suite of programs that is used to model and analyze systems, processes, or phenomena using simulation techniques. It allows users to create virtual representations of real-world systems and to experiment with these models in a safe and controlled environment.

Simulation software can be used to study a wide variety of systems, including manufacturing and logistics systems, energy systems, transportation systems, healthcare systems, and many others. It is particularly useful for studying complex systems that are difficult to analyze using traditional methods.

There are many different types of simulation software available, ranging from simple tools for creating simple models to complex software suites that can be used to build and analyze sophisticated simulations. Some examples of simulation software include Arena, Extendsim, Simio, AnyLogic, SimPy, NetLogo, Vensim, Simulink, System Dynamics, and GPSS.

Simulation software can be used to perform a variety of tasks, including:

- **Modeling:**
  Simulation software can be used to create virtual representations of real-world systems. These models can be as simple or as complex as needed and include a wide range of factors, such as the behavior of individual components, the interactions between components, and the influence of external factors.

- **Experimentation:**
  Once a model has been created, simulation software can be used to conduct experiments with the model. This can involve manipulating the model in various ways (e.g., changing input values or altering the behavior of

components) and observing the effects on the model's performance.

- **Analysis:**

  Simulation software can be used to analyze the results of experiments and to extract valuable insights and information from the data. This can involve calculating statistical quantities (e.g., means, standard deviations, confidence intervals), comparing the results of different experiments, and visualizing the data in various ways (e.g., using graphs, plots, or tables).

Selection of Simulation Software

When choosing simulation software, there are many features and considerations to take into account. Some of the things to consider when selecting simulation software include:

1. Do not focus on a single issue, such as ease of use. Consider the accuracy and level of detail obtainable, ease of learning, vendor support, and applicability to our applications.
2. Execution speed is important. Do not think exclusively in terms of experimental runs that take place at night and over the weekend. Speed affects development time. During debugging, an analyst might have to wait for the model to run up to the point in simulated time where an error occurs many times before the error is identified.
3. Beware of advertising claims and demonstrations. Many advertisements exploit the positive features of software only. Similarly, demonstrations solve the test problem very well, but perhaps might not be efficient for our problem.
4. Ask the vendor to solve a small version of the problem that is being simulated.
5. Determine whether the simulation package and language are sufficiently powerful to avoid having to write the logic in any external language.
6. Beware of "no programming required," unless either the package is a near-perfect fit to your problem domain, or

programming is possible with the supplied blocks, nodes, or process flow diagram.

Simulation Tools

Simulation tools are software programs that are used to create and run models of systems to study and analyze their behavior. Simulation tools can be used to model a wide range of systems, including computer systems, manufacturing systems, transportation systems, and many other types of systems.

There are many different types of simulation tools available, ranging from simple, general-purpose tools to specialized, domain-specific tools. Some examples of simulation tools include:

- **General-purpose simulation software:**
  These tools are designed to be used for modeling and simulating a wide range of systems. Examples include Arena, AnyLogic, and Simulink.

- **Hardware simulation tools:**
  These tools are designed specifically for modeling and simulating hardware systems, such as computer systems or manufacturing equipment. Examples include SPICE, VHDL, and Verilog.

- **Software simulation tools:**
  These tools are designed specifically for modeling and simulating software systems, including applications, operating systems, and other software. Examples include UML tools like Rational Rose and simulation software packages like Simula.

- **Network simulation tools:**
  These tools are designed specifically for modeling and simulating networked systems, including communication networks, transportation networks, and other types of systems. Examples include NS-2, NetSim, Packet Tracer, etc.

In addition to these types of specialized simulation tools, there are also many other types of tools that can be used for modeling and

simulating specific types of systems. The appropriate simulation tool for a given application will depend on the specific needs and goals of the simulation, as well as the type and complexity of the system being modeled.

9.1 Simulation in Java

Java is a widely available programming language that has been used extensively in simulation. It does not, however, provide any facilities directly aimed at aiding the simulation analyst, who therefore must program all details of the event-scheduling/time-advance algorithm, the statistics-gathering capability, the generation of samples from specified probability distributions, and the report generator. Nevertheless, the runtime library does provide a random-number generator. Unlike FORTRAN or C, the object-orientedness of Java does support modular construction of large models. For the most part, the special-purpose simulation languages hide the details of event scheduling, whereas in Java all the details must be explicitly programmed. To a certain extent though, simulation libraries such as SSF alleviate the development burden by providing access to standardized simulation functionality and by hiding low-level scheduling minutiae.

The following components are common to almost all models written in Java:

- **Clock**: A variable defining simulated time
- **Initialization method:** A method to define the system state at time t = 0
- **Min-time event method:** A method that identified the imminent event, i.e., elements of future event list (FutureEventList) that have the smallest timestamp
- **Event methods:** For each event type, a method to update the system state (and cumulative statistics) when that event occurs.
- **Random-variate generators:** Method to generate samples for a desired probability distribution.
- **Main program:** To maintain overall control of the event-scheduling algorithm.

- **Report generator:** A method that compares summary statistics from cumulative statistics and points report at end of the simulation.

The simulation begins by setting the Simulation clock to zero, initializing cumulative statistics to zero generating any initial event and placing them on Future Event List, and defining the system state at time t=0. The simulation program then cycles, repeatedly passing the current least time event to appropriate event methods until the simulation is over. The overall structure of Java simulation is:

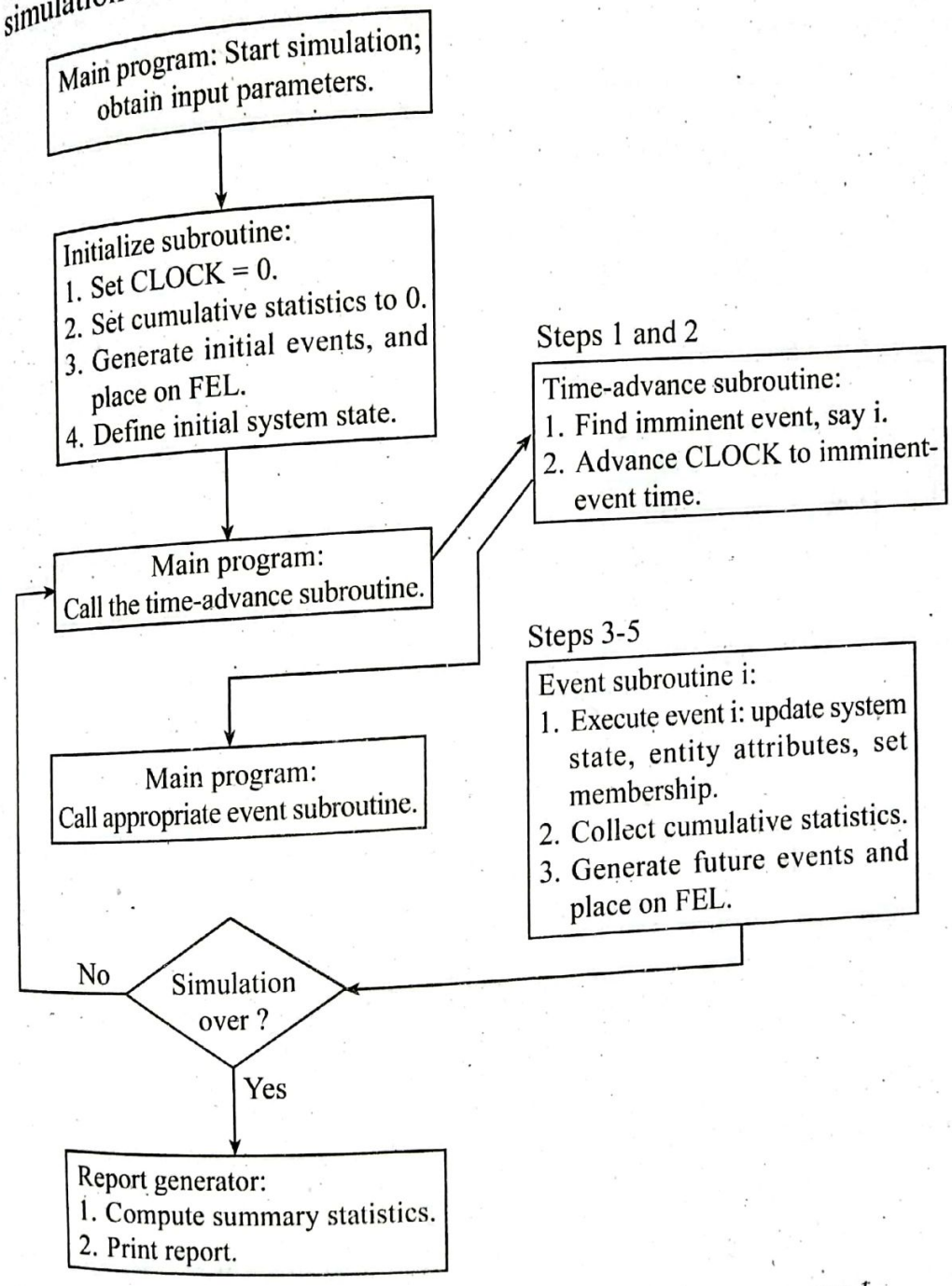


Fig.: Overall structure of a Java simulation program.[1]

Example: Single-Server Queue Simulation in Java

Let us consider a grocery checkout counter a typical single server queue, being simulated in Java which serves up to 1000 customers. Interarrival time is distributed exponentially with a mean of 4.5 minutes, the service time is distributed normally with a mean of 3.2 minutes and $\sigma = 0.6$ minutes. This system contains mainly 2 events: arrival and departure and its simulation contains the essential components of discrete event simulation.

The following flowchart shows all methods for event scheduling of a single server queue.

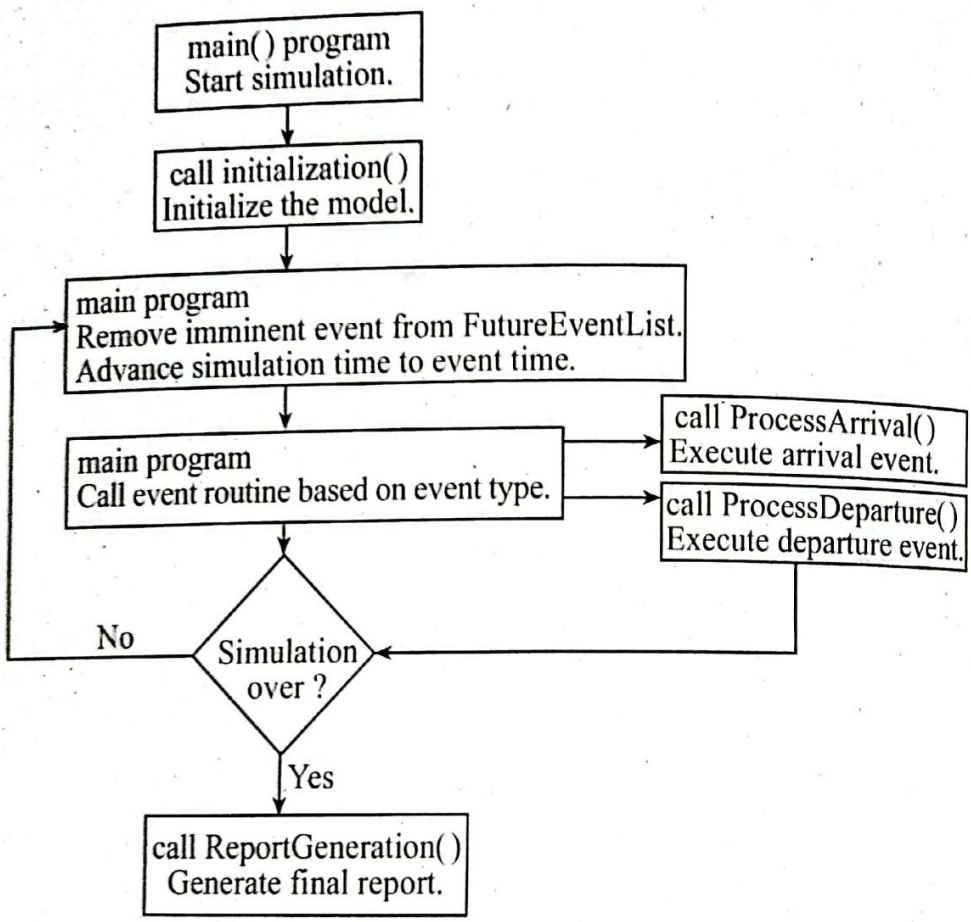


Fig.: Overall structure of Java simulation of a single server queue.[2]

The class Event stores a code for the event type (arrival or departure) and event time - stamp. It has the associated methods (functions) for creating an event and accessing its data. It also has associated methods that compare the event with another and then report whether the first event should be considered to be smaller,

equal, or larger than the argument event. The methods for this single server model and flow of control are shown in the above flowchart.

This could be described in Java as follows:

The entry point of the program and location of the control logic is through the class sim. Variables of classes Event List and Queue are declared. A variable of the Java built-in class Random is also declared; instances of this class provided random number streams. The main method controls the overall flow of the event-scheduling/time advance algorithm.

Structure of the main program:

```
class Sim {

// Class Sim variables
public static double Clock, MeanInterArrivalTime,
        MeanServiceTime, SIGMA, LastEventTime,
        TotalBusy, MaxQueueLength, SumResponseTime;
public static long NumberOfCustomers, QueueLength,
        NumberInService, TotalCustomers, NumberOfDepartures,
        LongService;

public final static int arrival = 1;
public final static int departure = 2;

public static EventList FutureEventList;
public static Queue Customers;
public static Random stream;
```

```
public static void main(String argv[]) {

        MeanInterArrivalTime = 4.5; MeanServiceTime = 3.2;
        SIGMA = 0.6; TotalCustomers = 1000;
        long seed =1000; //Long.parseLong(argv[0]);

        stream = new Random(seed);  // initialize rng stream
        FutureEventList = new EventList();
        Customers = new Queue();

        Initialization();
        // Loop until first "TotalCustomers" have departed

        while(NumberOfDepartures < TotalCustomers) {
                Event evt = (Event) FutureEventList.getMin(); //
                get imminent event
                FutureEventList.dequeue(); // be rid of it
                Clock = evt.get_time(); // advance simulation time
                if(evt.get_type() == arrival) ProcessArrival(evt);
                else ProcessDeparture(evt);
                }
        ReportGeneration();
}
```

Output:

The output after the completion of the simulation is presented below. The output can vary with each run due to the presence of some randomness in the system.

Single server queue simulation - grocery store checkout Counter

Mean interarrival time	4.5
Mean service time	3.2
Standard deviation of service times	0.6
Number of customers served	1000

Server utilization	0.7175
Maximum line length	7.0
Average response time	6.7358 Minutes
Proportion who spend four minutes or more in system	0.675
Simulation runlength	4455.02 Minutes
Number of departures	1000

Note: Most of the output statistics are estimates that contain random errors.

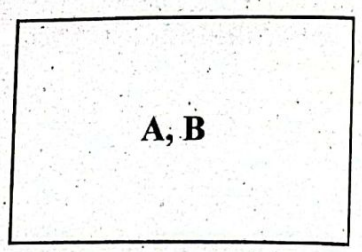
9.2 Simulation in GPSS

General Purpose Simulation System (GPSS) is a simulation software package that is used to model and analyze discrete-event systems. Discrete-event systems are systems in which the behavior of the system is determined by a sequence of discrete events that occur at specific points in time. GPSS is particularly well-suited for modeling and analyzing systems that involve the processing and movement of discrete items, such as jobs, parts, or packages. GPSS was developed in the 1960s and has been widely used in a variety of industries, including manufacturing, logistics, and healthcare. It is known for its simplicity and ease of use, making it a popular choice for students and practitioners who are new to simulation.

GPSS models are created using a specialized programming language that is specifically designed for building discrete-event simulations. The programming language consists of a set of commands that are used to specify the behavior of the system being modeled. GPSS models can be run using a simulation engine, which executes the model and generates a trace of the simulation. The trace can be used to analyze the results of the simulation and to extract useful insights and information from the data.

In GPSS symbols are used to represent different types of entities in the simulation model. Some common symbols used in GPSS include:

1. ADVANCE



An ADVANCE block delays the progress of a Transaction for a specified amount of simulated time.

Syntax: ADVANCE A, B

Operands:

A - The mean time increment. Required.

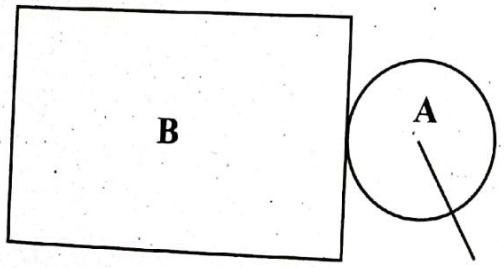
B - The time half-range or, if a function, the function modifier. Optional.

Example:

ADVANCE 101.6, 50.3

This example creates a block which chooses a random number between 51.3 and 151.9, inclusively (i.e. 101.6 plus or minus 50.3), and delays the entering Transaction that amount of simulated time.

2. DEPART



A DEPART block registers statistics which indicate a reduction in the content of a Queue Entity.

Syntax: DEPART A, B

Operands:

A - Queue Entity name or number. Required.

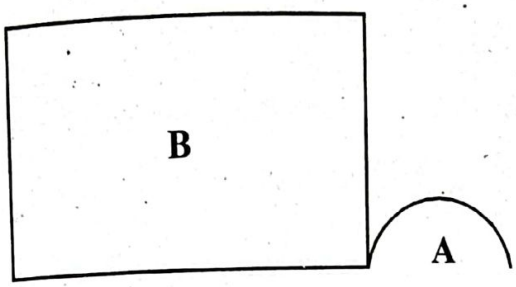
B - Number of units by which to decrease content of the Queue Entity. Default value is 1. Optional.

Example:

DEPART WaitingLine

In this example, the content of the Queue Entity named WaitingLine is reduced by one and the associated statistics accumulators are updated.

3. **ENTER**



When a Transaction attempts to enter an ENTER block, it either takes or waits for a specified number of storage units.

Syntax: ENTER A, B

Operands:

A - Storage Entity name or number. Required.

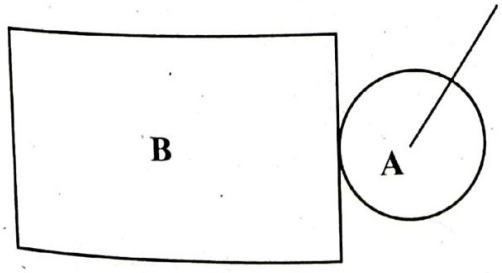
B - Number of units by which to decrease the available storage capacity. Default value is 1. Optional.

Example:

ENTER Toolkit, 2

In this example, the Active Transaction demands 2 storage units from the storage units available at the Storage Entity named Toolkit. If there are not enough storage units remaining in the Storage Entity, the Transaction comes to rest on the Delay Chain of the Storage Entity.

4. **QUEUE**



A QUEUE block updates Queue Entity statistics to reflect an increase in content.

Syntax: QUEUE A, B

Operands:

A - Queue entity name or number. Required.

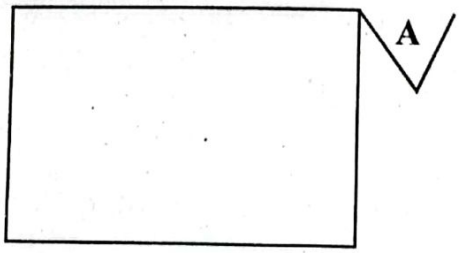
B - Number of units by which to increase the content of the Queue Entity. Default value is 1. Optional.

Example:

QUEUE WaitingLine

In this example, the content of the Queue Entity named WaitingLine is increased by one and the associated statistics accumulators are updated.

5. RELEASE



A RELEASE block releases ownership of a Facility, or removes a preempted Transaction from contention for a facility.

Syntax: RELEASE A

Operand:

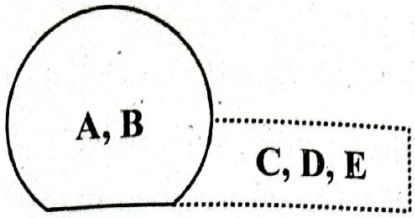
A - Facility number. Required.

Example:

RELEASE Teller1

In this example, when a Transaction which owns the Facility Entity named Teller1 enters the RELEASE Block, it gives up ownership to the Facility.

6. **GENERATE**



A GENERATE block creates Transactions for future entry into the simulation.

Syntax: GENERATE A, B, C, D, E

Operands:

A - Mean inter-generation time. Optional.

B - Inter-generation time half-range or Function Modifier. Optional.

C - Start delay time. Time increment for the first Transaction. Optional.

D - Creation limit. The default is no limit. Optional.

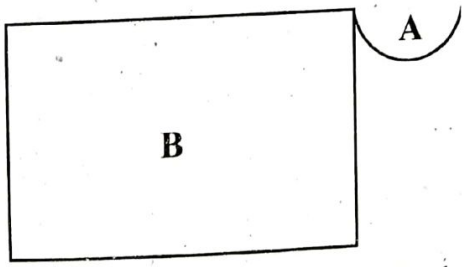
E - Priority level. Optional. Zero is the default.

Example:

GENERATE 0.1

This is the simplest way to use the GENERATE block. This Block causes a priority zero Transaction to enter the simulation every tenth of a time unit.

7. **LEAVE**



A LEAVE block increases the accessible storage units at a storage entity.

Syntax: LEAVE A, B

Operands:

A - Storage Entity name or number. Required.

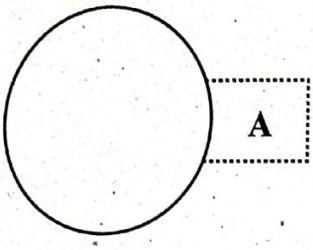
B - Number of storage units. The default is 1. Optional.

Example:

LEAVE RepairMen, 10

In this example, when a Transaction enters the LEAVE Block, the available storage units at the Storage Entity named RepairMen is increased by 10.

8. TERMINATE



A TERMINATE block removes the Active Transaction from the simulation and optionally reduces the Termination Count.

Syntax: TERMINATE A

Operand:

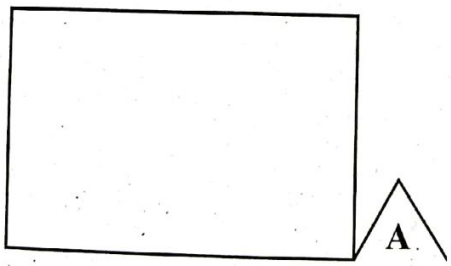
A - Termination count decrement. Default is 0. Optional.

Example:

TERMINATE 1

In this example, when a Transaction enters the TERMINATE block it is removed from the simulation. Also, the termination count of the simulation, which is set by a START command is decremented by 1.

9. SEIZE



When the Active Transaction attempts to enter a SEIZE block, it waits for or acquires ownership of a Facility Entity.

Syntax: SEIZE A

Operand:

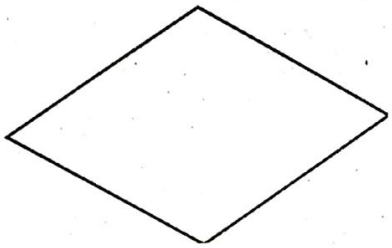
A - Facility name or number. Required.

Example:

SEIZE Teller1

In this example, when a Transaction attempts to enter the SEIZE Block, the state of the Facility named Teller1 is tested. If it is idle, ownership is given to the Active Transaction, which is allowed to enter the SEIZE Block and proceed to the Next Sequential Block (NSB). If the Facility is busy (owned), the Active Transaction comes to rest on the Delay Chain of the Facility.

10. **TRANSFER**



A TRANSFER block causes the Active Transaction to jump to a new block location.

Syntax: TRANSFER A, B, C

Here, the control jumps to location C with probability A, jumps to location B with probability 1 – A.

These are some examples of the symbols that are used in GPSS. There are many other symbols available, each of which serves a specific purpose in the simulation.

Examples of GPSS: Manufacturing Shop Model Simulation

A machine tool is producing parts at a rate of 1 per every 5 minutes. As they are finished, the parts goes to an inspector, who takes 4±3 minutes to examine each one and rejects about 10 % of the parts.

Simulate the system using the GPSS model.

Description:

A GENERATE block is used to represent the output of a machine by creating a transaction every 5 units of time. An ADVANCE block with a mean of 4 and modifier 3 represents inspection. So the time of inspection might be 4±3 (i.e. 1, 2, 3, 4, 5, 6, 7); with equal probability to each value. After completion, the transaction goes to TRANSFER with a selection factor of 0.1, representing rejection probability. Hence, 90% of the part goes to the next location i.e., ACCEPT indicating the accepted parts and 10% goes to REJECT, which represents rejected transactions. And both locations end with the TERMINATE block. Both TERMINATE blocks have 1; so, the terminating counter will add up the total number of transactions that terminate including both good and bad part.

Code:

```
        GENERATE 5, 0                    ; Create Parts
        ADVANCE 4, 3                     ; Inspect
        TRANSFER 0.1, ACCEPT, REJECT ; ACCEPT or
                                         ;REJECT
ACCEPT TERMINATE 1                       ; Accept Parts
REJECT  TERMINATE 1                      ; Reject Parts
```

Flow diagram:

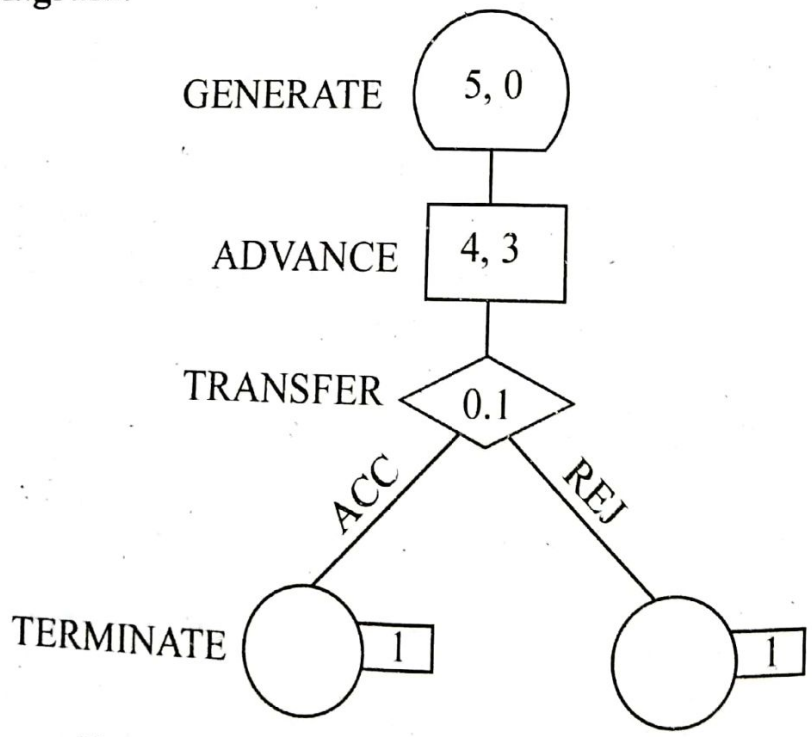


Fig.: GPSS model of shop manufacturing

9.3 Simulation in SSF

SSF (Scalable Simulation Framework) is a software platform that provides a single, unified interface for discrete-event simulation. It is an API (Application Programming Interface) that describes a set of capabilities for object-oriented and process view simulation and is available for C++ and Java. SSF is designed to achieve high performance, and defines five base classes: *Event, Entity, In Channel, Out Channel, and Process*. The Process class implements a thread of control, in which the action method contains the execution body. The Entity class describes simulation objects, while the In Channel and Out Channel classes describe the communication endpoints. The Event class defines messages sent between entities. SSF bridges the gap between models developed in pure Java and models developed in languages specifically designed for simulation. It also provides the flexibility offered by a general-programming language and has essential support for simulation.

Here are the steps involved in the SSF simulation process:

- **Starting the simulation:** A simulation starts when any entity's startAll() method is called, usually from the main() routine. The caller specifies the simulation's start time (defaults to 0) and end time.
- **Initialization:** After startAll() has been called, the framework calls the init() routines of all processes and entities. The Entity method now() should return the simulation start time during initialization.
- **Process execution:** All processes become eligible to run at the start time. Their actual execution is controlled by the framework, which executes the process's action() callback method.
- **Framework inner loop:** The framework provides nonprimitive scheduling of processes. Processes must eventually suspend themselves by issuing wait on() or wait

for() calls to allow the simulation clock to advance. The simulation time advances based on the arrival of events on channels and the duration of the wait for() statements.

- **Start, pause, resume, and join:** The startAll(), pauseAll(), resumeAll(), and joinAll() methods may not be called from within process code. pauseAll() allows the simulation to be paused gracefully, and resumeAll() resumes it. At any point, a call to joinAll () (e.g., from main ()) will block without the possibility of pause or resumption until the simulation execution is complete.

- **Framework concurrency:** Each entity is aligned to some object. Processes that are eligible to execute at each instant of simulation time will be scheduled by the framework: sequentially within an alignment group and concurrently across alignment groups.

- **Simulation time:** The behavior of an SSF model is defined by the collective actions of its processes on its state. Each modification of the model state takes place at a particular simulation time.

Example:

Here are the main components of the Single Server Queue System (Checkout Counter) simulation in the SSF:

- **SSQueue class:** This class contains the entire simulation experiment. It defines the experimental constants, contains SSF communication endpoints, and defines the inner class arrival.

- **Arrival process:** This is an SSF process that stores the identity of the entity, creates a random number generator, and enqueues the generated new arrivals. It then blocks the inter-arrival time.

- **Server process:** This process is called when a job has completed service or by a signal from the arrival process. It

also updates statistics. Customers are dequeued from the waiting list, or the process suspends if no customers are waiting.

9.4 Other Simulation Software

There are many different simulation software packages available, each with its unique features and capabilities. Here are a few examples:

- **Arena:** A powerful and user-friendly simulation software package that is widely used in manufacturing, logistics, and other industries.
- **AnyLogic:** A multi-method simulation software that supports a variety of simulation approaches, including discrete event, agent-based, and system dynamics.
- **Extendsim:** A powerful simulation software package that is particularly well-suited for modeling complex systems with multiple variables and interactions.
- **Simio:** A simulation software package that is designed to be easy to use, with a powerful drag-and-drop interface and a wide range of built-in models and components.
- **Simul8:** A simulation software package that is widely used in healthcare, manufacturing, and other industries. It includes a powerful graphical interface and a wide range of built-in models and components.

There are many other simulation software packages available as well, each with its unique features and capabilities. In general, it is a good idea to carefully consider your simulation needs and goals before choosing a particular software package.

There are many different factors that you should consider when choosing a simulation software package, including the complexity of your simulation model, the type of simulation approach you need (e.g., discrete event, agent-based, system dynamics), the level

of support and documentation available, and the overall cost of the software.

It is generally a good idea to try out several different simulation software packages to see which one best meets your needs. Some software packages, such as Arena and AnyLogic, offer free trial versions that you can download and use to get a sense of the software's capabilities and ease of use. Other software packages may offer demo versions or online tutorials that allow you to see how the software works without actually downloading and installing it.

In general, it is a good idea to carefully consider your simulation needs and goals before choosing a particular software package. Some software packages may be more suitable for modeling complex systems with many variables and interactions, while others may be better suited for simpler models. Some software packages may offer more advanced features, such as the ability to run simulations on distributed computing systems, while others may be more focused on ease of use and user-friendliness. Ultimately, the best simulation software for you will depend on your specific needs and goals.

REFERENCES

[1] Jerry Banks et al. Discrete Event System Simulation. Fifth edition. Pearson Education Limited, 2014, p.94.

[2] Jerry Banks et al. Discrete Event System Simulation. Fifth edition. Pearson Education Limited, 2014, p.95.

1. **Imagine a machine tool in a manufacturing shop is turning out parts at the rate of one every 5 minutes. As they are finished the parts go to an inspector and the inspector takes 4±3 minutes to examine each one and rejects about 10% of parts. Develop the GPSS block diagram for the above scenario using the necessary GPSS blocks.** [2076 Bhadra]

Solution:

GPSS block diagram for given scenario is given below:

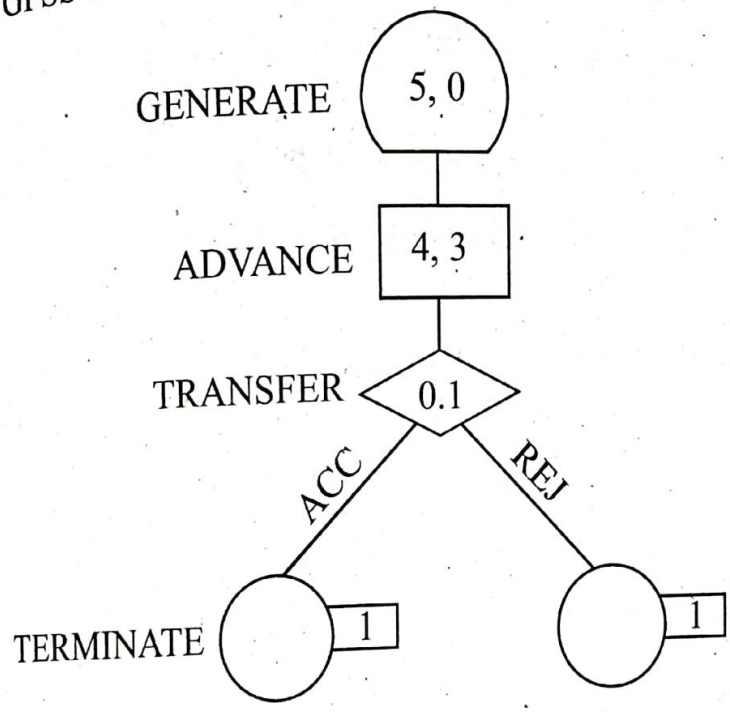


Fig.: GPSS model for manufacturing shop.

Exercise

1. Explain the overall structure of the Java simulation of a single server queue. [2080 Shrawan, 2073 Magh]
2. Explain simulation in GPSS with example. [2079 Chaitra]
3. What do you mean by simulation software? Explain the basic blocks of GPSS. [2079 Jestha]
4. Explain at least 4 GPSS block diagram symbols. [2078 Chaitra]
5. What is GPSS? Explain four generic blocks used in GPSS with examples. [2077 Chaitra]
6. Briefly explain simulation in Java. [2075 Bhadra]
7. Explain the structure of Java Simulation with an example. [2074 Bhadra]
8. Explain at least 5 GPSS block diagram symbols. [2074 Magh]
9. Explain the single server queuing simulation model using JAVA. [2073 Bhadra]
10. Explain the different features that are relevant when selecting simulation software. [2072 Magh]
11. Explain the simulation in SSF with an example. [2071 Magh]
13. Explain the GPSS model for a general scenario of a manufacturing shop with the conditional transfer. [2070 Magh]

CHAPTER 10

SIMULATION OF COMPUTER SYSTEMS

CHAPTER 10

SIMULATION OF COMPUTER SYSTEMS

Simulation of computer systems is the process of creating and running models of computer systems to study and analyze their behavior. This can be done using specialized simulation software, which allows users to design and build models of computer systems, specify inputs and scenarios, and run simulations to see how the system behaves under different conditions.

Simulation of computer systems is commonly used in academia and industry to study and understand the behavior of computer systems, evaluate different design choices and configurations, and predict the performance and reliability of systems under different conditions. It is a powerful tool for understanding and improving the design and operation of computer systems, and is used in a wide range of fields, including computer science, electrical engineering, and information technology.

Simulation of computer systems can be used to study a wide range of computer systems, including hardware systems, software systems, and networked systems. It allows researchers and designers to study and analyze the behavior of these systems in a controlled and systematic way and can provide valuable insights into their behavior and performance.

10.1 Level of Abstraction in Computer System

The *level of abstraction* in a computer system refers to the degree to which the system is represented and modeled in a simplified or idealized way. A higher level of abstraction means that the system is represented in a more simplified and abstract way, while a lower level of abstraction means that the system is represented more concretely and realistically.

Many different levels of abstraction can be used when modeling and simulating computer systems. Some common levels of abstraction include:

- **Computer system level:** At this level of abstraction, a computer system is modeled and simulated at a high level, including the hardware, software, and other components of the system. This level of abstraction may include modeling the overall architecture and design of the system, as well as the interactions between different components. For example, a computer system-level model might include representations of the processor, memory, and other hardware components, as well as the operating system, applications, and other software. This level of abstraction can be useful for studying the overall behavior and performance of a computer system, but may not capture all of the details and complexities of the system.
- **Processor level:** At this level of abstraction, the processor (or CPU) of a computer system is modeled and simulated at a medium level of abstraction. This level of abstraction may include modeling the microarchitecture and internal components of the processor, as well as the interactions between the processor and other components of the system. For example, a processor-level model might include representations of the processor's registers, caches, and other internal components, as well as the bus or other communication channels between the processor and other components such as memory or I/O devices. This level of abstraction can be useful for studying the behavior and performance of the processor itself, as well as its interactions with other components of the system.
- **CPU level:** At the CPU level, the processor (or CPU) of a computer system is modeled and simulated at a very low level of abstraction, including the internal circuits and components of the processor. Some of the components that might be included in a CPU-level model of a processor include:

- **Memory interface unit:** This component is responsible for managing the communication between the processor and the main memory of the computer system. It may include logic circuits for addressing and accessing memory locations, as well as buffers and other components for storing memory.
- **Translation look-aside buffer (TLB):** This component is a cache that is used to store recently accessed memory addresses and their corresponding physical memory locations. The TLB can improve the performance of the processor by allowing it to access memory locations more quickly, without having to perform a full memory address translation for each access.
- **Instruction prefetch unit:** This component is responsible for fetching instructions from memory and bringing them into the processor for execution. It may include logic circuits for accessing memory, as well as buffers and other components for storing instructions temporarily as they are fetched.
- **Microcode instruction unit:** This component is responsible for interpreting and executing the instructions that are fetched by the instruction prefetch unit. It may include logic circuits for decoding instructions, as well as control logic for executing the instructions on the processor's arithmetic and logic unit (ALU).
- **ALU:** This component is responsible for performing arithmetic and logical operations on data. It may include circuits for performing addition, subtraction, multiplication, division, and other operations, as well as logic circuits for performing logical operations such as AND, OR, and NOT.

- **Gate level:** At this level of abstraction, the individual logic gates and other components within a computer system are

modeled and simulated at the lowest level of abstraction. This level of abstraction is often used to model the behavior of very low-level circuits and components within a system, and can be very accurate but also very time-consuming and complex. For example, a gate-level model might include representations of individual transistors, resistors, and other components, as well as the connections between these components. This level of abstraction can be useful for studying the behavior of very low-level circuits and components, but may not be practical for larger or more complex systems.

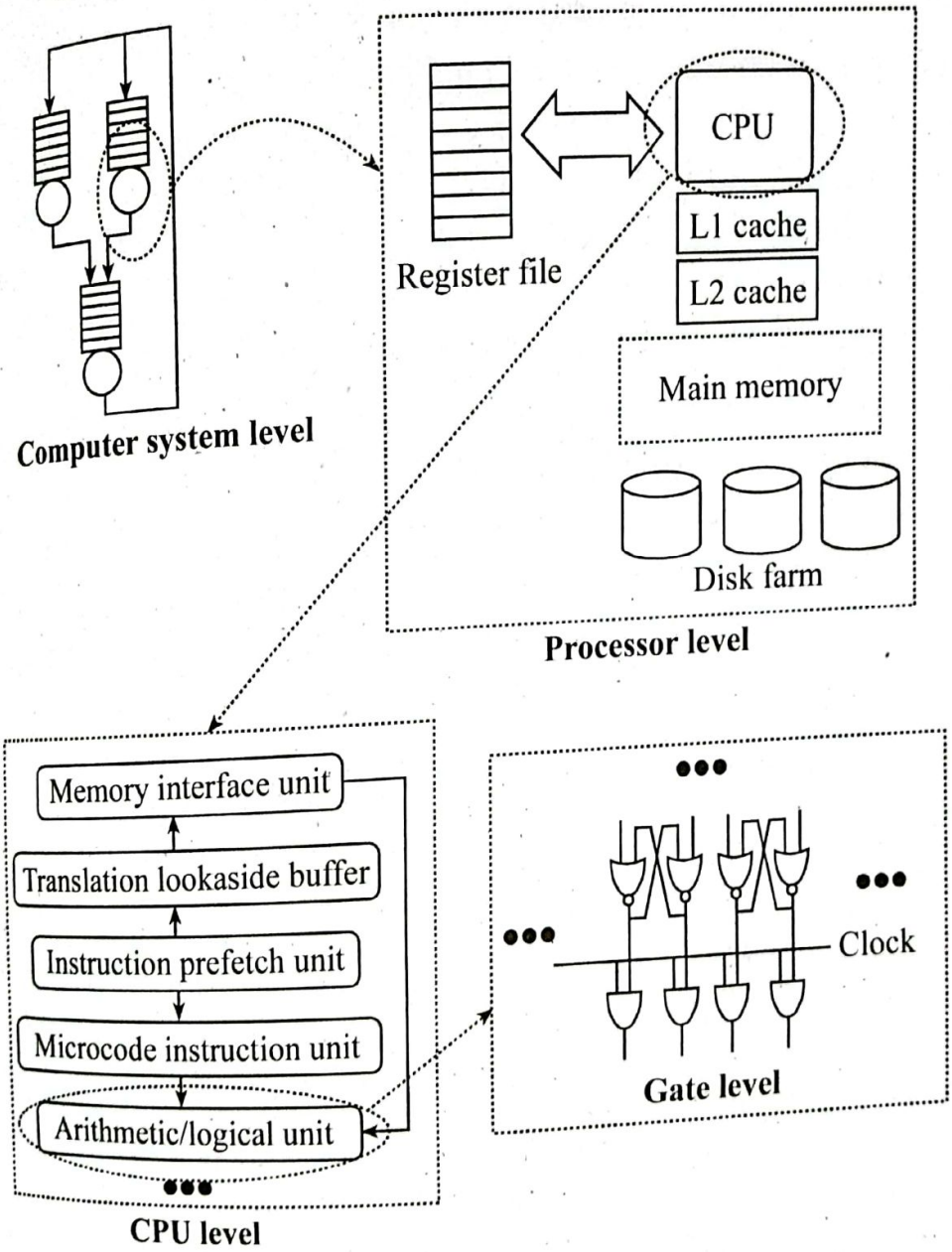


Fig.: Level of abstraction in computer system

10.2 High-Level Computer–System Simulation

High-level computer-system simulation can be useful for a variety of purposes, including analyzing the performance of a system under different workloads or conditions, evaluating the design of a system, or identifying bottlenecks or other issues that might affect the system's performance. It can also be used to study the effects of different design choices or changes to the system or to explore scenarios that might not be practical or feasible to test in the real world.

Example: Simulation of Web Servers (WWW)

To model and simulate the performance of a company's website and its associated data servers, application servers, and web servers during peak periods, it is necessary to focus on the impact of timing at each level of the system, as well as the factors that affect the timing and the effects of timing on contention for resources. The desired output of the simulation is the empirical distribution of the access response time for the website during peak periods.

To create this high-level simulation model, it will be necessary to represent each of the components of the system, including the data servers, application servers, web servers, and routers. The model should include representations of the communication channels and interfaces between these components, as well as models of external factors that might affect the system, such as user input or external devices.

In addition to modeling the components of the system, it will be necessary to consider the impact of timing on the system's performance. This might include analyzing the effects of different workloads or access patterns on the system, as well as the effects of contention for resources such as memory or processing power.

To run the simulation and collect data, it may be necessary to set up and run multiple simulations with different input and output parameters, to study the behavior and performance of the system under different conditions. Once the simulations are complete, the results can be analyzed to gain insights into the performance of the

system during peak periods and to identify any bottlenecks or other issues that might affect the system's ability to handle the load.

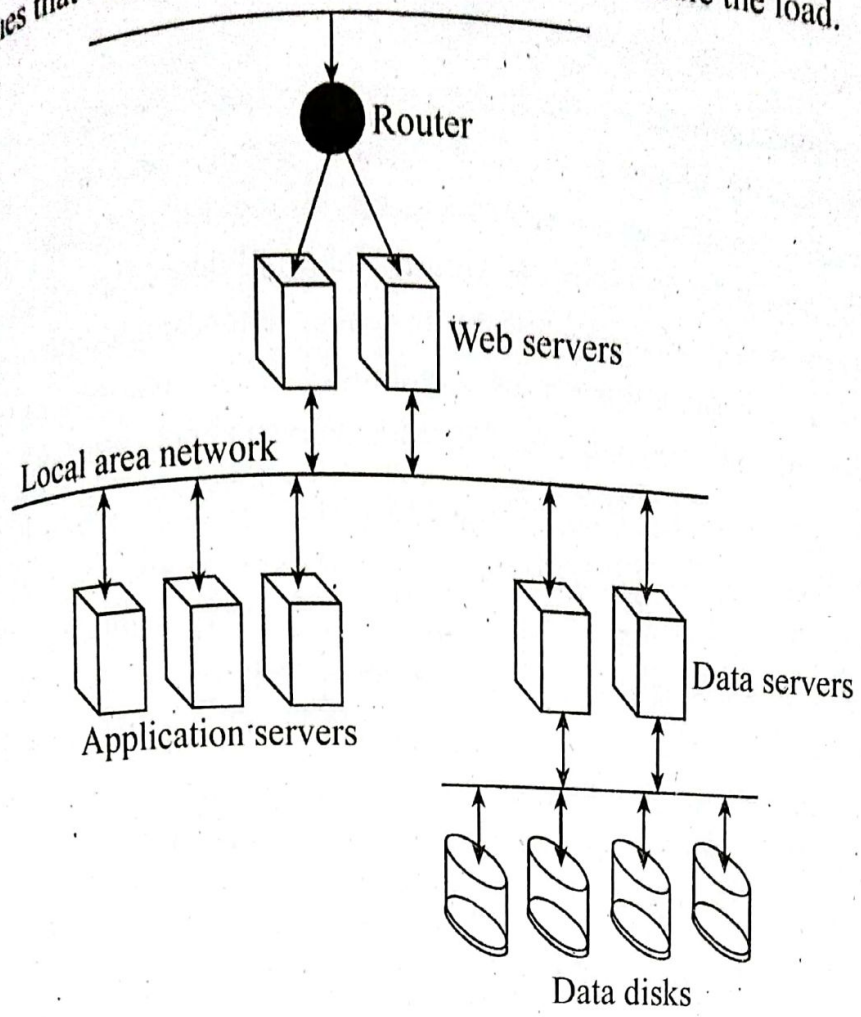


Fig.: Simulation of Web servers

Simulation Model

In this system, all entries into the system are made through a dedicated router. The router examines the request and forwards it to a web server, taking some time to decide whether the request is new or part of an ongoing session. The router outputs the selection of the web server and enqueues the request for service to the web server.

The web server consists of three queues: one for new requests, one for suspended requests that are waiting for a response from the application server, and one for requests that are ready to process the response from the application server. It is assumed that the web server has enough memory to handle all the requests and has a specific queuing policy. The web server also identifies the

associated application server for each new request and format and forwards the request for service to the application server, adding the request to the suspended queue.

The application server organizes service requests. A new request for service is added to the new request queue. Each application request is modeled as a sequence of sets of requests (organized in a burst) from data servers. It is assumed that all data requests from a burst must be satisfied before the next burst computation can begin. The application server maintains a list of ready-to-execute threads and a list of suspended threads for each application.

Data servers create a new thread to respond to data requests and place it in a queue of ready threads. When the data server receives a service request, the thread requests data from a disk and then places it in a suspended queue. The disk completes its operation for the data request, and the thread in the data server, upon receiving the response from the disk, moves to the ready list and reports back to the application server associated with the request.

The thread suspended at the application server responds and finishes, then reports its completion to the web server. The thread in the web server that initiated the request then communicates the results back to the internet.

Response Time Analysis

The distribution of query-response time can be estimated by measuring the time between when a request first hits the router and when the web server thread communicates the result. To analyze the system, it is necessary to measure the behavior at each server of each type. To assess the system's capacity at peak loads, it is possible to simulate the system to identify bottlenecks and then look for ways to reduce the load on bottleneck devices by changing various simulation settings such as the scheduling policy and queue discipline.

10.3 CPU Simulation

In *CPU simulation*, the focus is on discovering the execution time of the CPU. To perform CPU simulation, the input is a stream of instructions, and the simulation must model the logical design of what happens in response to the instruction stream. This includes modeling the behavior of the various components of the CPU, such as the instruction fetch unit, the instruction decode unit, the execution unit, and the memory interface unit. By simulating the execution of the instruction stream and measuring the resulting execution time, it is possible to analyze the performance of the CPU and identify any bottlenecks or other issues that might affect its efficiency. In addition to modeling the components of the CPU, it may also be necessary to consider external factors that might affect the performance of the CPU, such as the memory access patterns or the behavior of other devices in the system.

Problem Definition of ILP (Instruction Level Parallelism) CPU:

In an *ILP (Instruction-Level Parallelism) CPU*, there are several stages that an instruction goes through before it is executed. These stages include:

1. **Instruction fetch:** The instruction is fetched from memory.
2. **Instruction decode:** The memory word holding the instruction is interpreted to discover the operations to be performed and the registers involved.
3. **Instruction issue:** An instruction is issued if there are no constraints that would prevent it from being executed.
4. **Instruction execute:** The instruction operation is performed.
5. **Instruction complete:** The results of the instruction are stored in the destination register.
6. **Instruction graduate:** Executed instructions are graduated in the order that they appear in the instruction stream.

The ILP design allows for multiple instructions to be represented in some stages at the same time, which may cause the execution of

instructions out of order. The instruction graduate phase will reorder all the executed instructions to ensure that they are executed in the correct order. This allows for improved performance by allowing the CPU to execute multiple instructions concurrently.

Simulation Model of ILP (Instruction Level Parallelism) CPU:

The stages in an ILP (Instruction-Level Parallelism) CPU simulation can be described as follows:

1. **Instruction fetch:** In this stage, the instruction is fetched from the simulated memory system if present. If the memory system is present, it will look for the next referenced instruction in the instruction cache, stalling if there is a cache miss. This stage also places the instruction in the CPU's list of active instructions.

2. **Instruction decode:** In this stage, instruction is placed in the list. Physical registers are assigned to the logical registers that appear as the target of an operation. Registers used as operands are assigned physical registers that define their values. Branch instructions are identified and their outcomes are predicted, and resources for the instruction execution are committed.

3. **Instruction issue:** In this stage, a decoded instruction is issued for execution if the values in its input registers are available and a functional unit needed to perform the instruction is available. This can be achieved by marking the registers and functional units as busy or pending. After the state is changed, the instruction waiting for that register or functional unit is reconsidered for the issue.

4. **Instruction execute:** In this stage, the actual operation intended by the instruction is performed.

5. **Instruction complete:** In this stage, the result of the instruction is deposited into a register or memory as specified in the instruction.

6. **Instruction graduate:** In this stage, the completed instruction is reordered in the same order as the instruction stream. This is simulated by using the sequence number of the next instruction to be graduated to determine the correct order.

Overall, these stages are used to simulate the execution of instructions in an ILP CPU, allowing for the improvement of performance by executing multiple instructions concurrently. The simulation must model the logical design of what happens in response to the instruction stream, and focus on discovering the execution time of the instructions.

10.4 Memory Simulation

Memory simulation is the process of modeling and analyzing the behavior of a computer's memory system to understand its performance and optimize its design. Memory simulation is typically used in the design and development of computer systems to evaluate the impact of different memory configurations and to identify bottlenecks and potential issues that may arise in the system.

Several key components are typically included in a memory simulation model, including the memory hierarchy, the memory controller, and the memory bus. The memory hierarchy includes various levels of memory, such as the main memory, cache memory, and virtual memory, and models how data is stored and accessed at each level. The memory controller is responsible for managing the flow of data between the memory and the CPU, and the memory bus is the communication channel that connects the memory to the rest of the system.

Memory simulation can be performed using specialized software tools or through custom-built simulation models. These tools and

models allow engineers to analyze the performance of different memory configurations and optimize the design of the memory system for maximum efficiency and reliability.

In a computer system, memory is often organized in a hierarchical structure, with different levels of memory arranged in a specific order based on access speed and capacity. The memory hierarchy is designed to allow the CPU to access data as quickly as possible, by storing frequently used data in the faster levels of memory and less frequently used data in the lower levels.

The memory hierarchy typically consists of the following levels:

- **Register:**

  Register is a small, high-speed storage unit within the CPU that serves as a temporary storage location for frequently accessed data and instructions. It operates at the same speed as the CPU, providing quick access and improving overall efficiency. Register memory has a limited size and capacity, with different types of registers serving specific purposes. The CPU uses instructions to read from and write to registers, perform operations, and transfer data. Register allocation techniques aim to optimize the use of available registers. Overall, register memory plays a crucial role in enhancing CPU performance by facilitating fast data access and manipulation. Registers are part of CPU itself.

  L1 Cache: This is the fastest level of memory in the hierarchy, and it is located directly on the CPU. L1 cache is typically small in size, but it has a very high access speed, allowing the CPU to quickly retrieve data from it.

  L2 Cache: L2 cache is slightly slower than the L1 cache, but it is larger in size and can store more data. It is often located on the motherboard or within the CPU itself.

- **Main Memory (RAM):**

  Main memory, or random-access memory (RAM), is the primary working memory of the computer. It is used to store data and instructions that the CPU is currently processing.

Main memory is relatively fast, but it is slower than cache memory.

Disks:

Disks, such as hard drives and solid state drives, are the slowest level of memory in the hierarchy. They are used to store long-term data that is not currently being used by the CPU.

The memory hierarchy allows the CPU to access data quickly by first checking the faster levels of memory, such as L1 and L2 cache, before falling back to the slower levels, such as main memory and disks. This helps to ensure that the CPU can access the data it needs as efficiently as possible, improving the overall performance of the system.

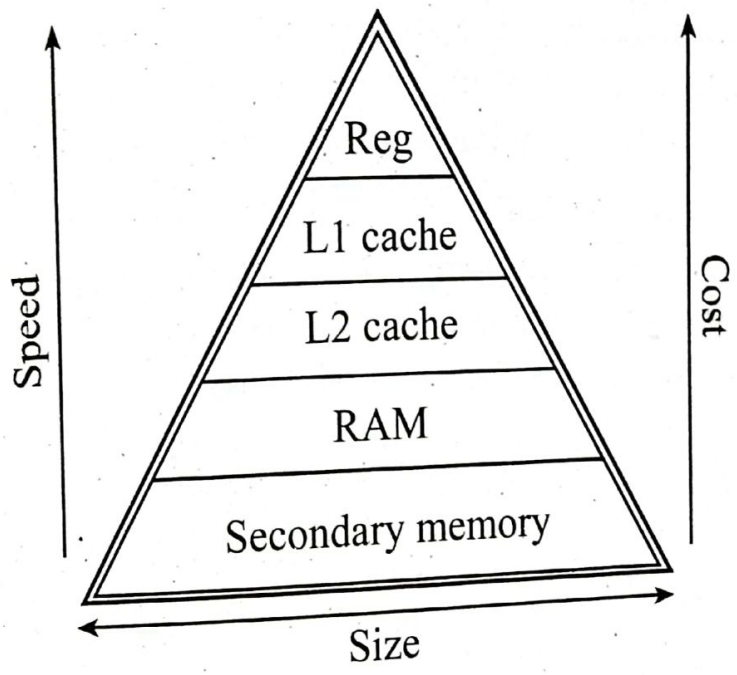


Fig.: Memory hierarchy

Example: Cache Simulation

Cache simulation involves modeling the behavior of cache memory in a computer system. This can be useful for understanding how different design choices, such as cache size and associativity, impact the performance of the system.

To simulate cache behavior, one can create a model of a cache memory system and input a stream of memory accesses to the model. The model can then simulate the behavior of the cache,

tracking things like hit rate (the percentage of memory accesses that are served by the cache) and miss rate (the percentage of memory accesses that require a trip to the main memory).

Several factors can be varied in a cache simulation:

- **Cache size:**
  The size of the cache can be varied to see how it impacts performance. A larger cache size may result in a higher hit rate, as more data can be stored in the cache, but it also consumes more physical space and may be more expensive to implement.
- **Cache associativity:**
  Cache associativity refers to the number of cache lines that can be mapped to a particular memory address. Higher associativity may result in a higher hit rate, as it allows more data to be stored in the cache, but it may also require more complex hardware and consume more power.
- **Replacement policy:**
  The cache replacement policy determines how the cache handles situations where a new memory access requires a cache line to be evicted (removed) to make room for the new data. Different replacement policies, such as least recently used (LRU) and first in, first out (FIFO), can be compared to see how they impact performance.

To perform the simulation on cache we can create a cache directory which is a data structure used to manage the contents of a cache memory system. It is typically implemented as an array, with each entry in the array representing a directory entry. The purpose of the cache directory is to keep track of which cache lines are currently stored in the cache and to maintain the least recently used (LRU) status of those lines.

When a memory access is made to the cache, the cache directory is used to determine whether the data being accessed is present in the cache. If the data is present, it is considered a cache hit and the cache can serve the request directly. If the data is not present, it is

considered a cache miss and the cache must fetch the data from the main memory.

In the event of a cache miss, the cache directory is updated to reflect the new contents of the cache. This may involve evicting (removing) an existing cache line to make room for the new data, depending on the cache's replacement policy. The LRU status of the cache lines within the set is also updated at this time.

Maintaining the cache directory and LRU status of the cache lines is an important aspect of cache simulation. It helps to accurately model the behavior of the cache and understand how different design choices impact performance.

Cache simulation can be useful for understanding how different design choices impact the performance of a cache memory system and can help designers make informed decisions about how to optimize the design of their systems.

Exercise

1. Explain the simulation model of a computer system that requests the services from World Wide Web (WWW). [2080 Shrawan, 2073 Magh]
2. Explain CPU simulation with an example. [2079 Chaitra, 2075 Bhadra, 2074 Bhadra]
3. Explain different levels of abstraction to be considered in the simulation of a computer system. [2079 Jestha, 2078 Chaitra]
4. Discuss memory simulation with examples. [2074 Magh]
5. Explain the simulation of a central processing unit for the lower level of abstraction. [2073 Bhadra]
6. Explain CPU simulation by sketching a simulation model of the computer system. [2071 Bhadra]

BIBLIOGRAPHY

Jerry Banks et al. *Discrete Event System Simulation*. Fifth edition. Pearson Education Limited, 2014.

Averill M. Law. *Simulation Modeling and Analysis*. Fifth edition. McGraw-Hill Education, 2015

Geoffrey Gordon. *System Simulation*. Second edition. Prentice Hall of India Private Limited, 2008.

Mark W. Spong. *Introduction to Modeling and Simulation: A Systems Approach*. Wiley, 2023

Manuel D. Rossetti. *Simulation Modeling and Arena*. Third edition, 2021

https://ww3.ticaret.edu.tr/mckasapbasi/files/2013/10/SSME_QueueingTheory.pdf

https://sites.pitt.edu/~dtipper/2130/2130_Slides5.pdf