Unit-2: Simulation of Continuous and Discrete System

Continuous System

In simulation and modeling, a continuous system refers to a system whose state evolves continuously over time, as opposed to discrete systems where state changes occur at distinct, separate points in time. Continuous systems are often described using differential equations or other mathematical models that involve continuous variables.

Examples:

- **Mechanical Systems:** Such as the motion of a pendulum, the behavior of a spring-mass-damper system, or the dynamics of a vehicle moving along a road.

- **Electrical Systems:** Such as circuits with resistors, capacitors, and inductors, where voltages and currents vary continuously with time.

- **Chemical Systems:** Such as reactions occurring in a solution, where concentrations of reactants and products change continuously.

- **Biological Systems:** Such as population dynamics, where the growth or decline of a population occurs continuously.

Continuous System Models:

Many systems comprise computers, communications networks, and other digital systems to monitor and control physical (electrical, mechanical, thermodynamic, etc.) processes. Models of these systems have some parts modeled as discrete event systems, other parts modeled with continuous (differential or differential-algebraic) equations, and the interaction of these parts is crucial to understanding the system's behavior.

Following points describes the concept of continuous system:

- In continuous simulation, continuously changing state variables of a system are modeled by differential equations.
- A continuous system is the system in which the activities of the main elements of the system cause smooth changes in the attributes of the entities of the system.
- On mathematical modeling, the attributes of the system are controlled by the continuous functions.
- In such system, the relationships depicts the rates at which the attributes changes.
- The continuous system is modeled using the differential equations.
- The complex continuous system with non-linearity can be simulated by showing the application to models for linear differential equations to obtain constant coefficients and then generalize to more complex equations.

Analog Computer:

An analog computer is a type of computer that processes information using continuously variable physical quantities, such as voltage, current, and pressure. These quantities can take on any value within a given range and are represented and processed using analog circuits and devices, such as amplifiers, resistors, and capacitors.

Analog computers are unified with devices like Integrator, Differentiator, Adder, Inverter, Multiplier, and Divider so as to simulate the continuous mathematical model of the system, which generates continuous outputs.

Analog computers are good at solving problems that involve continuous mathematical functions, such as differential equations and simulations of physical systems. They can provide real-time solutions to complex problems, as the computations are performed by continuously varying the physical quantities.

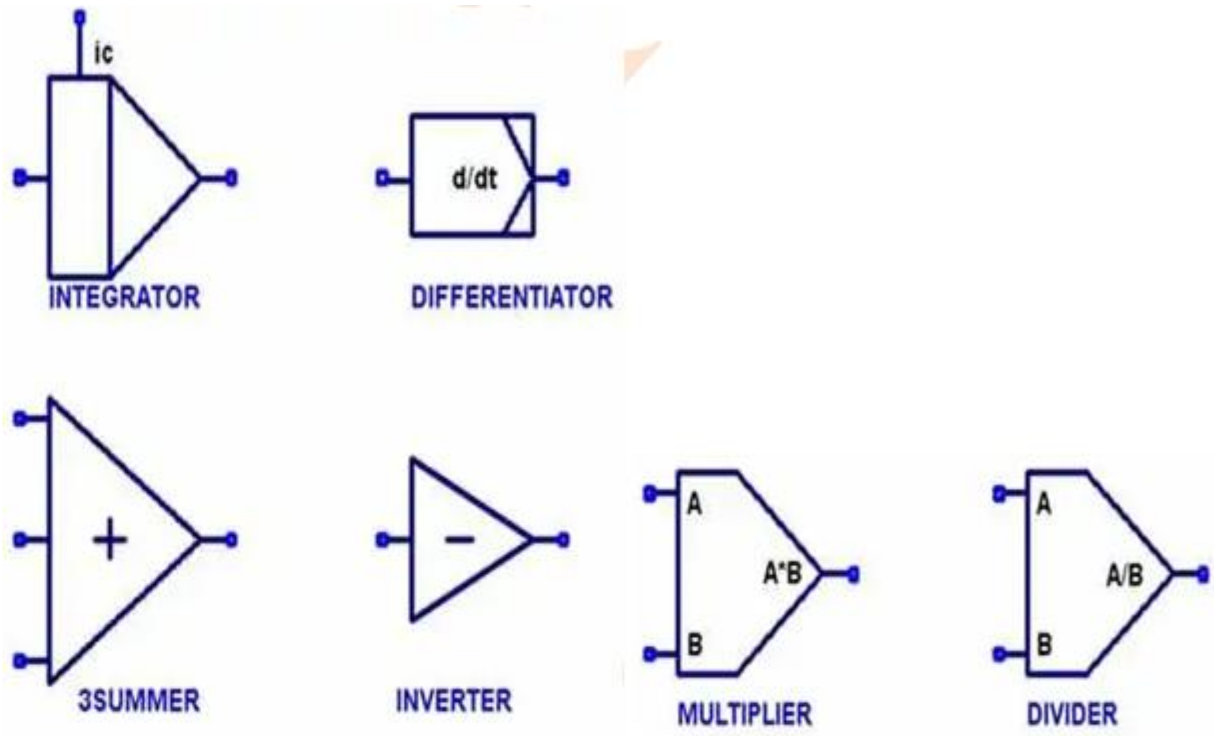
Analog computers were used for control systems, scientific simulations, and military applications. However, the development of digital computers has made them less common today.

Analog method of system simulation is for use of analog computer and other analog devices in the simulation of continuous system. The analog computation is sometimes called differential analyzer.

Analog Methods:

Analog method refers to a way of solving mathematical problems by directly representing physical quantities using continuously varying physical quantities, such as voltage, current, or rotation angle.

Following are the analog symbols used to design the analog computer.



Question:
Design analog computer of……

1) $\boldsymbol{M\ddot{x} + D\dot{x} + kx = kF(t)}$

Solution:

The general method by which analog computers are applied can be demonstrated using second order differential equation.

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

Solving the equation for the highest order derivate gives,
$M\ddot{x} = kF(t) - D\dot{x} - kx$

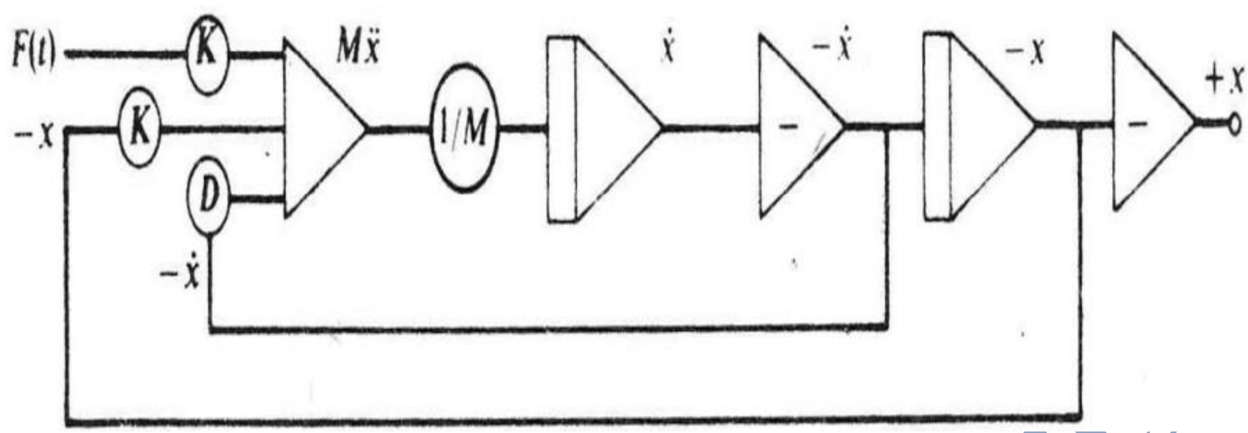


Fig: Analog simulation of automobile suspension system

Suppose a variable representing the input f(t) is supplied, assume there exists variables representing $-x$ and $-\dot{x}$. These three variables can be scaled and added to produce $M\ddot{x}$. Integrating it with a scale factor $1/M$ produces $\dot{x}$. Changing sign produces $-\dot{x}$, further integrating produces $-x$, a further sign inverter is included to produce $+x$ as output.

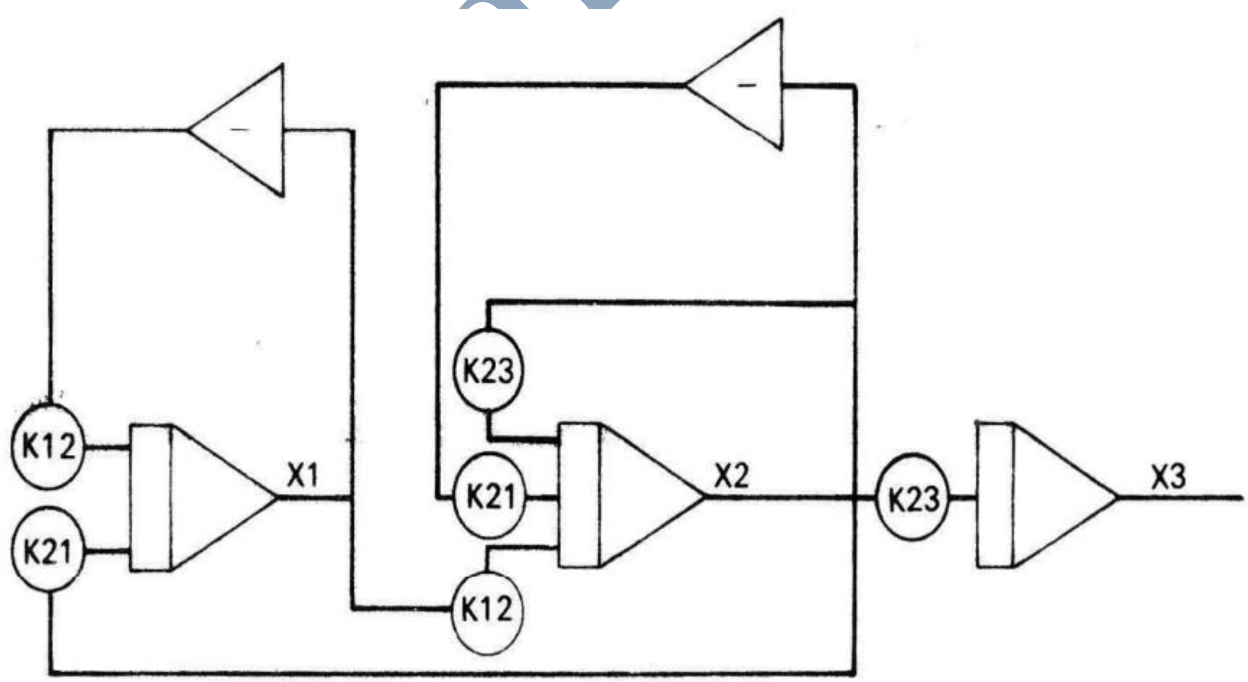
When a model has more than one independent variable, a separate block diagram is drawn for each independent variable and where necessary, interconnections are made between the diagrams.

Example: Design analog computer of

$x'_1 = -k12x1 + k21x2$

$x'_2 = k12x1 - (k21 - k23)\ x2$

$x'_3 = k23x2$



There are three integrators. Reading from left to right, they solve the equations for $x1$, $x2$ & $x3$. Interconnections between the three integrators with sign changers where necessary provides inputs that define the differential coefficients of the three variables.
First integrator, for example is solving the equation,
$x'_1 = -k12x1 + k21x2$

The second integrator is solving the equation
$x'_2 = k12x1 - (k21 - k23)2$
$= k12x1 - k21x2 + k23x2$

In this case, the variable $x2$ is being used twice as an input to the integrator, so that the two coefficients $k21$ and $k23$ can be changed independently. The last integrator solves the equation
$x'_3 = k23x2$

Assignment:

Q1) Design analog computer of
$\ddot{x1} - k\dot{x1} + k1\dddot{x1} = a\ x3$
$\dddot{x2} + k2x1 - k1\ x3 - k3x2 = 0$
$\dddot{x3} - k2x2 + \ddot{x1} = f(x)$
Q2) Design analog computer of
$\dot{y1} - \dot{x1} + k1\ \dddot{z1} = 0$
$\ddot{x1} = x1 - \dot{x1}z1$
$\dddot{z1} = k2x1 + \dddot{x1}$
Q3) Design analog computer of
$\dot{x} - k\dot{x} + k\dddot{x} = x$
$\dot{y} = x - k\ z - k3x$
$\dddot{z} = k\ y + \dot{z}$

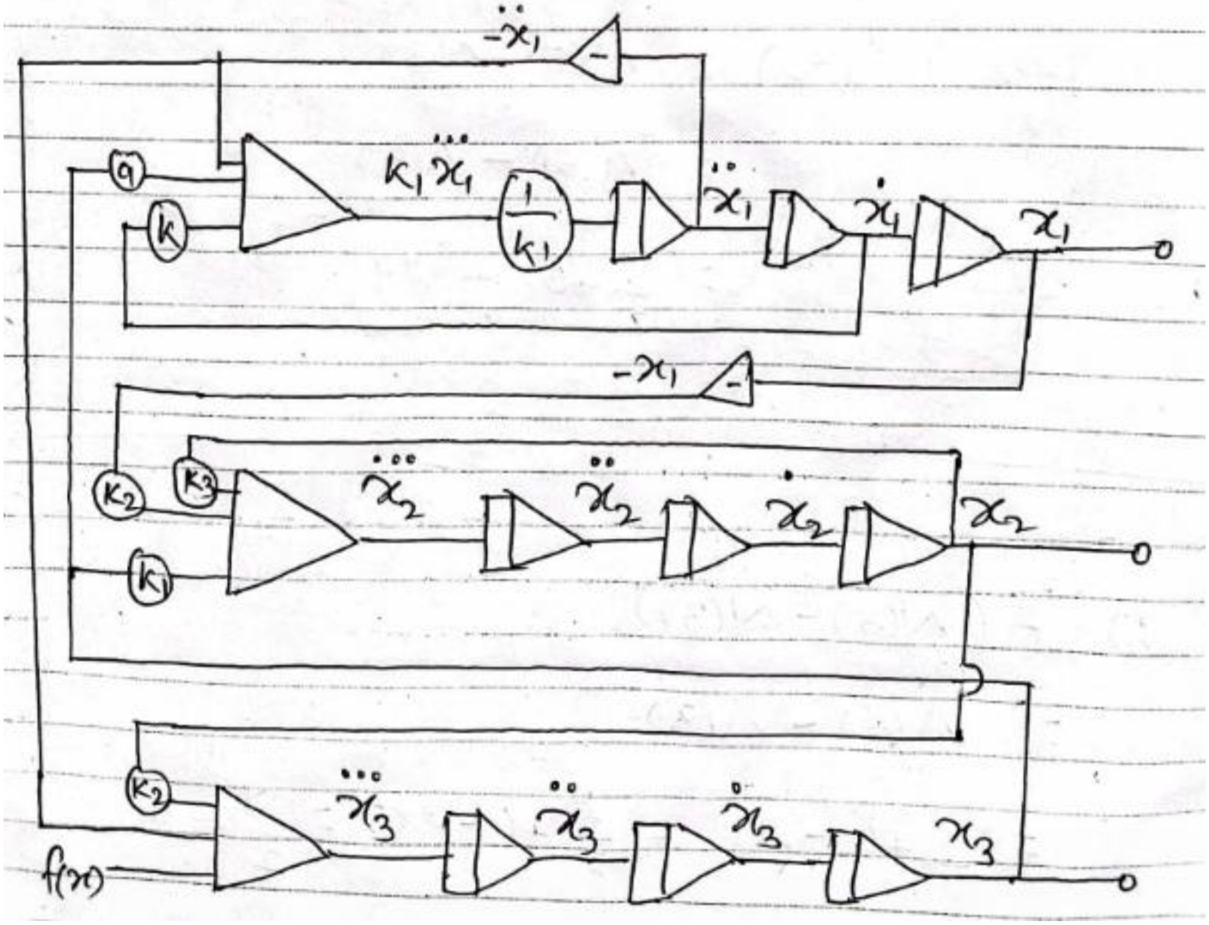
Solution of Assignment- Analog Computer Design

Q1) Solution:-

$$k_1 \dddot{x}_1 = a x_3 - \ddot{x}_1 + k \dot{x}_1$$

$$\dddot{x}_2 = k_1 x_3 + k_3 x_2 - k_2 x_1$$

$$\dddot{x}_3 = f(x) + k_2 x_2 - \ddot{x}_1$$

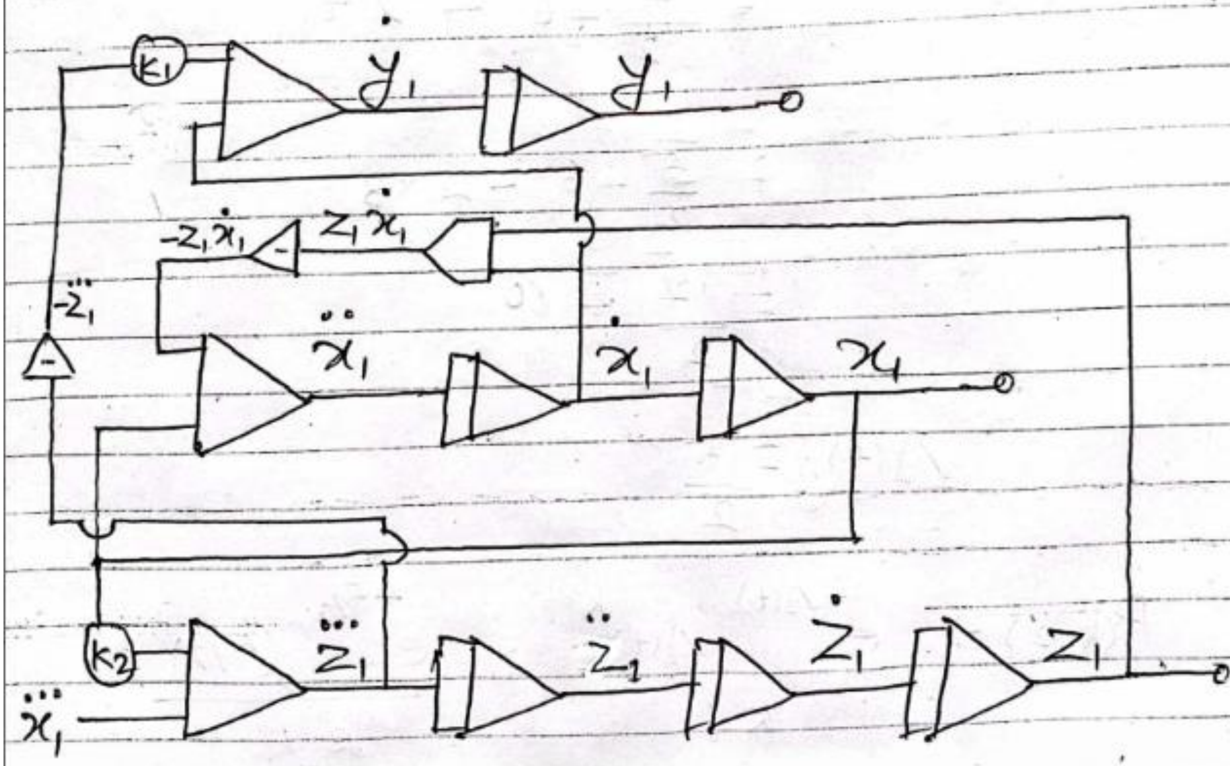


Q2) Solution

$$\dot{y}_1 = \dot{x}_1 - k_1 \dddot{z}_1$$

$$\ddot{x}_1 = x_1 - \dot{x}_1 z_1$$

$$\dddot{z}_1 = k_2 x_1 + \dddot{x}_1$$

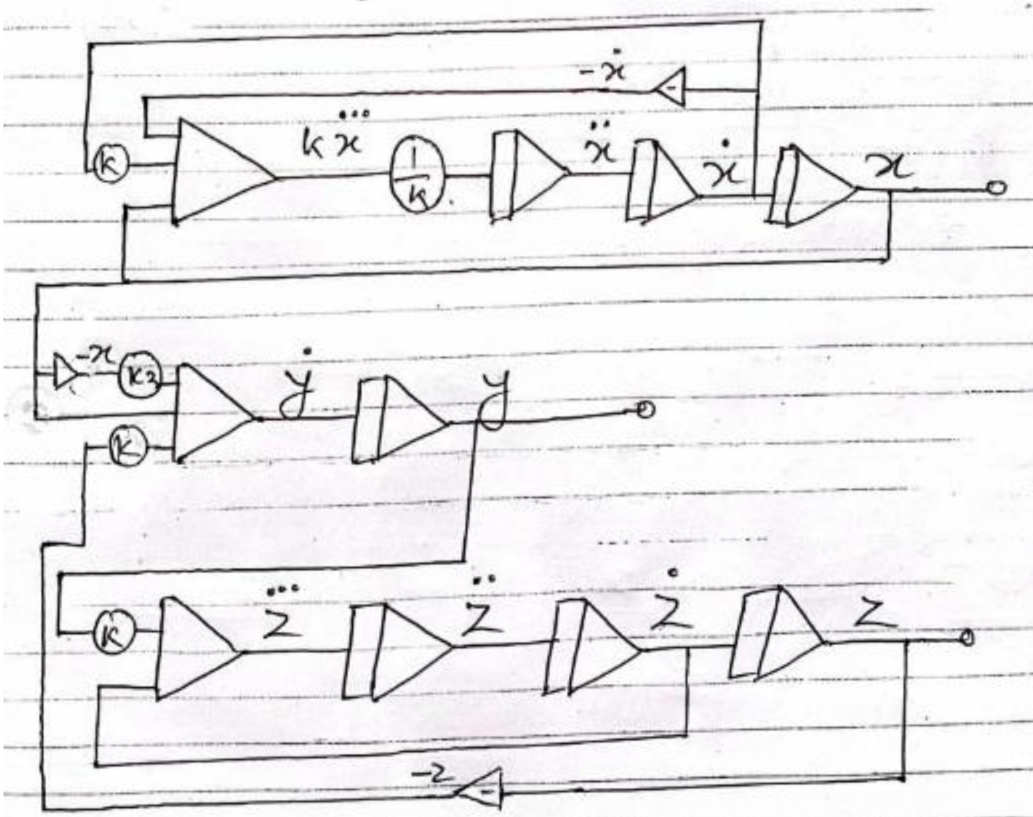


Q3) solution

$$k\dddot{x} = x - \dot{x} + k\dot{x}$$

$$\dot{y} = x - kz - k_3 x$$

$$\dddot{z} = ky + \dot{z}$$



Hybrid Simulation:

For most studies the model is clearly either of a continuous or discrete nature and that is the determining factor in deciding whether to use an analog or digital computer for system simulation. However, there are times when an analog and digital computers are combined to provide simulation. In this circumstances hybrid simulation is used. Hybrid simulation is provided by combining analog and digital computers. The form taken by hybrid simulation depends upon the applications.

Here, one computer may be simulating the system being studied while other is providing a simulation of the environment in which the system is to operate. It is also possible that the system being simulated is an interconnection of continuous and discrete subsystems, which can be modeled by an analog and digital computer being linked together.

The introduction of hybrid simulation required certain technological developments for its exploitation. High speed converters are needed to transform signals from one form of representation to the other.

Practically, the availability of mini computers has made hybrid simulation easier, by lowering costs and allowing computers to be dedicated to an application. The term "hybrid simulation" is generally reserved for the case in which functionality distinct analog and digital computers are linked together for the purpose of simulation.

Digital-Analog Simulators:

- To avoid the disadvantages of analog computers, many digital computer programming languages have been written to produce digital analog simulators.
- They 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 language contains macro instructions that carryout the actions of adders, integrators and sign changers.
- More powerful techniques of applying digital computers to the simulation of continuous systems have been developed.
- As a result, digital analog simulators are not now in expensive use.

Interactive System

Interactive systems are computer systems characterized by significant amounts of interaction between humans and the computer. Editors, CAD-CAM (Computer Aided Design-Computer Aided Manufacture) systems, and data entry systems are all computer systems involving a high degree of human-computer interaction. Games and simulations are interactive systems. Web browsers and Integrated Development Environments (IDEs) are also examples of very complex interactive systems.

Feedback System:

The system takes feedback from the output i.e. input is coupled with output. A significant factor in the performance of many systems is that coupling occurs between the input and output of the system. The term feedback is used to describe the phenomenon.

There are two types of feedback system:

- Negative feedback system
- Positive feedback system

Negative feedback system:

A negative feedback system is a regulatory mechanism in which the output or response of a process opposes the initial change that triggered it. In this system, when a particular event or signal occurs, it leads to a response that counteracts or reduces the intensity or magnitude of that event, restoring the system to its original state or equilibrium.

Example:

- **Thermostat-controlled heating system:**
  - Input: Desired room temperature set on the thermostat.
  - Output: Actual room temperature.
  - Feedback: If the room temperature exceeds the desired level, the thermostat reduces or shuts off the heating to bring it back to the set level.
- **Biological example:** Blood sugar regulation:
  - The pancreas releases insulin when blood sugar levels are high to bring them down. Once the levels normalize, insulin release decreases.

Hence…. in simple terms, negative feedback "feeds back" less of the output into the system, which helps to stabilize and control the process. It plays a crucial role in maintaining homeostasis and stability in various natural and artificial systems.

Negative feedback loops are commonly found in biological systems, such as the regulation of body temperature, blood pressure, and hormone levels. In these cases, when a certain parameter deviates from its set point, negative feedback mechanisms kick in to bring it back within the desired range.

Additionally, negative feedback is widely used in engineering and control systems to ensure stability and accuracy. By continuously monitoring the output and adjusting it in the opposite direction to the deviation, negative feedback helps maintain the desired operating conditions.

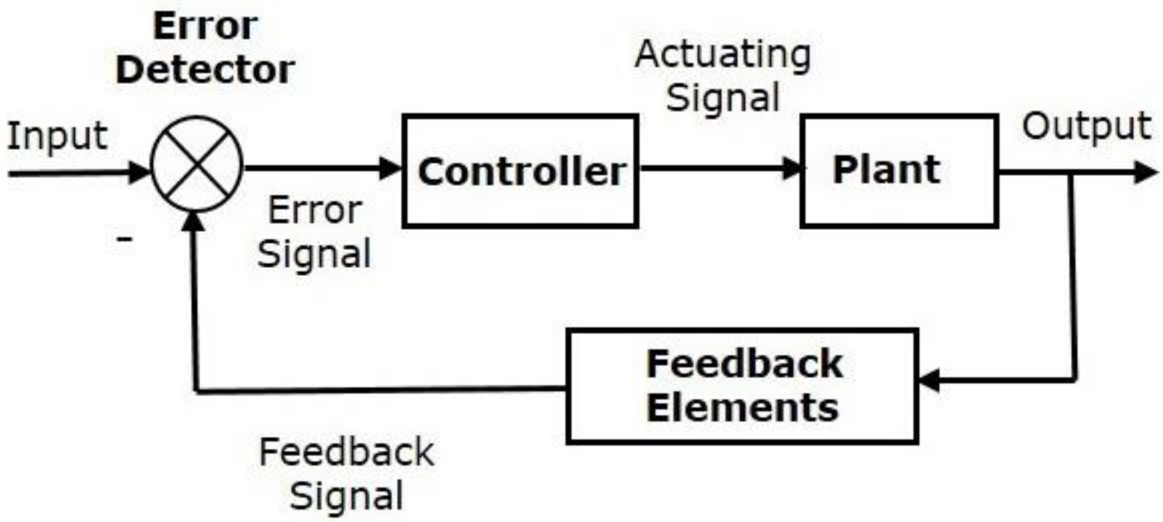


Fig: Negative feedback system

Positive feedback system:

A positive feedback system is a type of regulatory mechanism in which the output or response of a process amplifies or reinforces the initial change that triggered it. In this system, when a particular event or signal occurs, it leads to an increase in the intensity or magnitude of that event, resulting in further stimulation of the process.

Example:

- **Microphone and speaker loop:**
  When a microphone picks up sound from a speaker and feeds it back into the system, it creates a feedback loop that amplifies the sound, often resulting in a loud, high-pitched noise.
- **Biological example: Childbirth contractions:**
  During childbirth, the release of oxytocin increases uterine contractions, which in turn stimulates more oxytocin release, intensifying the contractions until the baby is born.
- **Snowball Effect (Avalanche Formation)**
  As a small snowball rolls down a snowy slope, it gathers more snow, becoming larger and heavier. This increase in size and weight enables it to gather even more snow, amplifying its growth and momentum.
- **Population Growth (Exponential Growth in Biology)**
  In ideal conditions, as a population grows, the number of individuals capable of reproduction increases. This results in an even greater number of offspring, causing exponential population growth.

In simple terms, positive feedback "feeds back" more of the output into the system, promoting its continuation and amplification. This can lead to exponential growth or escalation in certain situations. Positive feedback loops can be found in various natural and artificial systems, including biology, economics, climate, and engineering.
It's important to note that while positive feedback can lead to dramatic changes, it can also make systems more unstable and prone to reaching extremes. As a result, some systems have built-in mechanisms to counteract or dampen positive feedback loops to maintain stability. On the other hand, negative feedback systems work in the opposite way, where the response opposes the initial change and helps to maintain equilibrium or balance in the system.

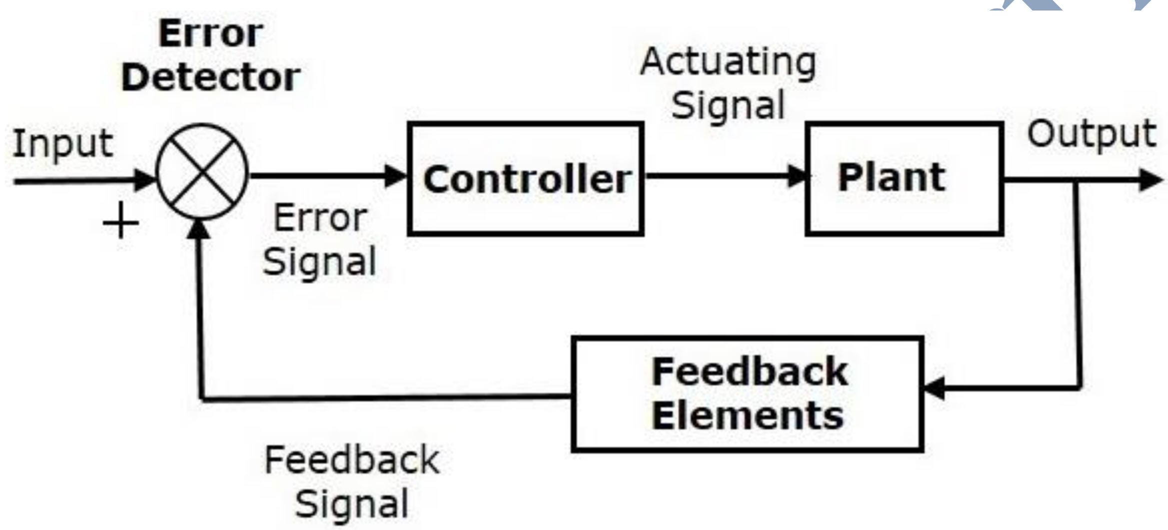


Fig: Positive feedback system

Example;

- Autopilot aircraft system
- A home heating system controlled by a thermostat
- Error Correction mechanism

Autopilot aircraft System:

In the aircraft feedback system, the input is a desired aircraft heading and the output is the actual heading. The gyroscope of the autopilot is able to detect the difference between the two headings (desired and actual output). Feedback is established by using the difference to operate the control surface. Since change of heading will then affect the signal being used to control the heading.
The difference between the desired signal θ_t and actual heading θ_0 is called the error signal, since it is a measure of the extent to which the system from the desired condition. It is denoted by ϵ.

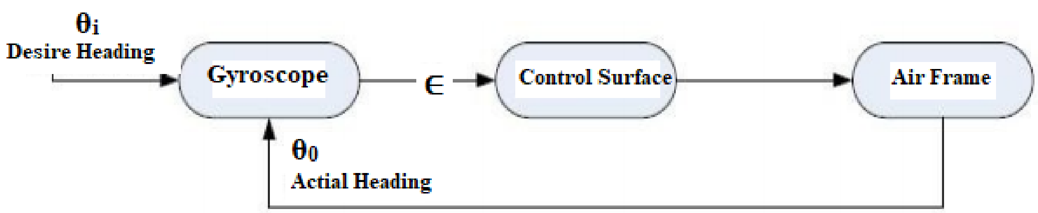


Fig: An aircraft under autopilot control (Continuous system)

In order to simulate the action of autopilot, we first construct a mathematical model of the aircraft system. The error signal, **ϵ**, has been defined as the difference between the desired heading, or output, θ_i, and the actual heading, θ_0. We therefore have the following identity:

$$\varepsilon = \theta_i - \theta_0 \qquad (1)$$

We assume the rudder (a flat, movable piece usually of wood or metal that is attached to a ship, boat, airplane, etc., and is used in steering) is turned to an angle proportional to the error signal, so that the force changing the aircraft heading is proportional to the error signal. Instead of moving the aircraft sideways, the force applies a torque which will turn the aircraft.

$$\text{Torque} = K\varepsilon - D\dot{\theta}_0 \qquad (2)$$

Where K and D, are constants, the first term on the right-hand side is the torque produced by the rudder, and the second is the viscous drag.

The torque is also given by the product of inertia on body of aircraft and second derivative of the heading,

$$\text{Torque} = I\ddot{\theta}_0 \qquad (3)$$

From equation (1), (2), and (3) we get,

$$I\ddot{\theta}_0 + D\dot{\theta}_0 + K\theta_0 = K\theta_i \ (4)$$

Dividing both sides by I, and making the following substitutions in equation (4)

$$\ddot{\theta}_0 + \frac{D\dot{\theta}_0}{I} + \frac{K\theta_0}{I} = \frac{K\theta_i}{I} \ (4)$$

$$2\xi\omega = D/I,$$

$$\omega^2 = k/I$$

We get……

$$\ddot{\theta}_0 + 2\xi\omega\dot{\theta}_0 + \omega^2\theta_0 = \omega^2\theta_i \qquad (5) \ \ (\xi \text{ is damping factor})$$

This is a second order differential equation.

Note: Torque is the measure of the force that can cause an object to rotate about an axis.

Graphical representation of autopilot aircraft is shown in figure bellow:

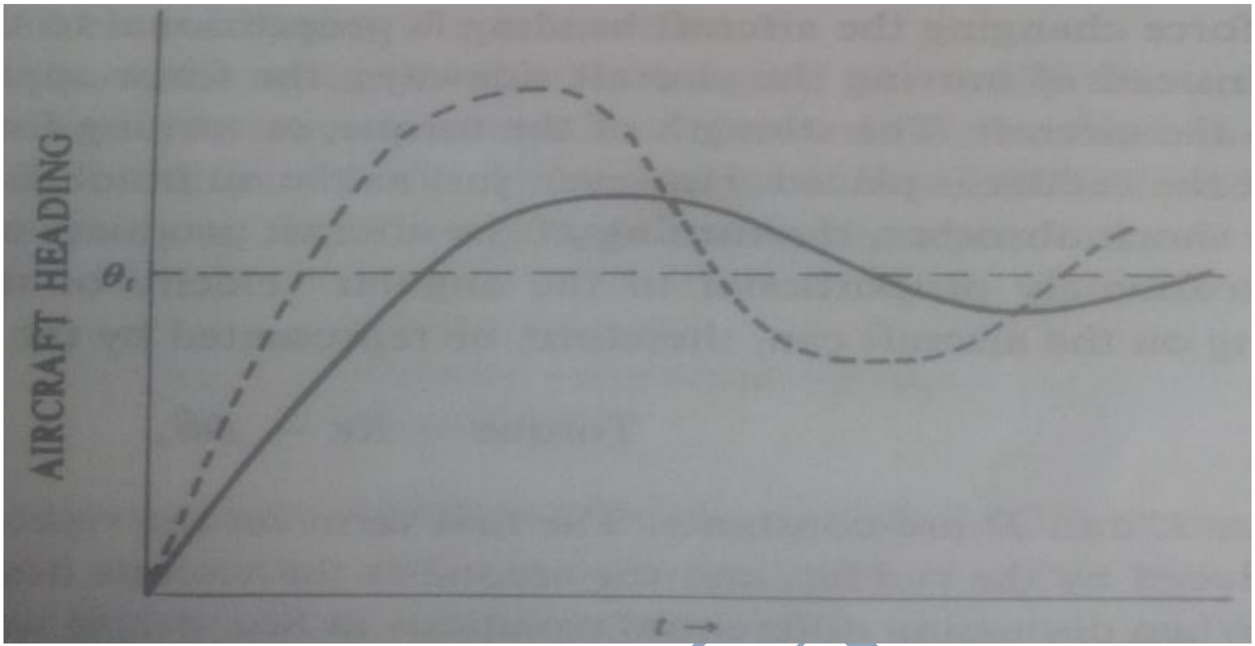


Fig: Aircraft response to autopilot system

Home heating system:

The system has a furnace whose purpose is to heat a room, and the output of the system can be measured as a room temperature. Depending upon whether the temperature is below or above the thermostat setting, the furnace will be turned on or off, so that information is being feedback from the output to input. In this case there are only two states either the furnace is on or off.

Discrete-Event Simulation:

Discrete-Event Simulation (DES) is a modeling technique used to simulate systems where the state changes at distinct points in time due to the occurrence of discrete events. Each event represents a specific change in the system's state, and the simulation progresses by jumping from one event to the next, rather than continuously. DES can be either deterministic or stochastic, depending on the nature of the target process.

Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation time can directly jump to the occurrence time of the next event, which is called **next-event time progression**.

In addition to next-event time progression, there is also an alternative approach, called **incremental time progression**, where time is broken up into small time slices and the system state is updated according to the set of events/activities happening in the time slice. Because not every time slice has to be simulated, a next-event time simulation can typically run faster than a corresponding incremental time simulation.
Both forms of DES contrast with continuous simulation in which the system state is changed continuously over time on the basis of a set of differential equations defining the rates of change of state variables.

Example:
A common exercise in learning how to build discrete-event simulations is to model a Queuing, such as customers arriving at a bank teller to be served by a clerk (*bank staff*).
In this example, the system objects are ***Customer*** and ***Teller***, while the system events are ***Customer-Arrival***, ***Service-Start*** and ***Service-End***. Each of these events comes with its own dynamics defined by the following event routines:

1. When a ***Customer-Arrival*** event occurs, the state variable ***queue-length*** is incremented by 1, and if the state variable ***teller-status*** has the value "**available**", a ***Service-Start*** follow-up event is scheduled to happen without any delay, such that the newly arrived customer will be served immediately.
2. When a ***Service-Start*** event occurs, the state variable ***teller-status*** is set to "**busy**" and a ***Service-End*** follow-up event is scheduled with a delay (obtained from sampling a ***service-time*** random variable).
3. When a ***Service-End*** event occurs, the state variable ***queue-length*** is decremented by 1 (representing the customer's departure). If the state variable ***queue-length*** is still greater than zero, a ***Service-Start*** follow-up event is scheduled to happen without any delay. Otherwise, the state variable ***teller-status*** is set to "**available**".

The random variables that need to be characterized to model this system stochastically are the ***inter-arrival-time*** for recurrent ***Customer-Arrival*** events and the ***service-time*** for the delays of ***Service-End*** events.

The attributes of discrete-event simulation:

At a high-level, discrete-event simulation is built on top of the following components:

- **System** – a collection of entities with certain attributes.
- **State** – a collection of attributes representing the system's entities.
- **Event** – an occurrence in time that may alter the system's state.

Components of Discrete-Event Simulation:

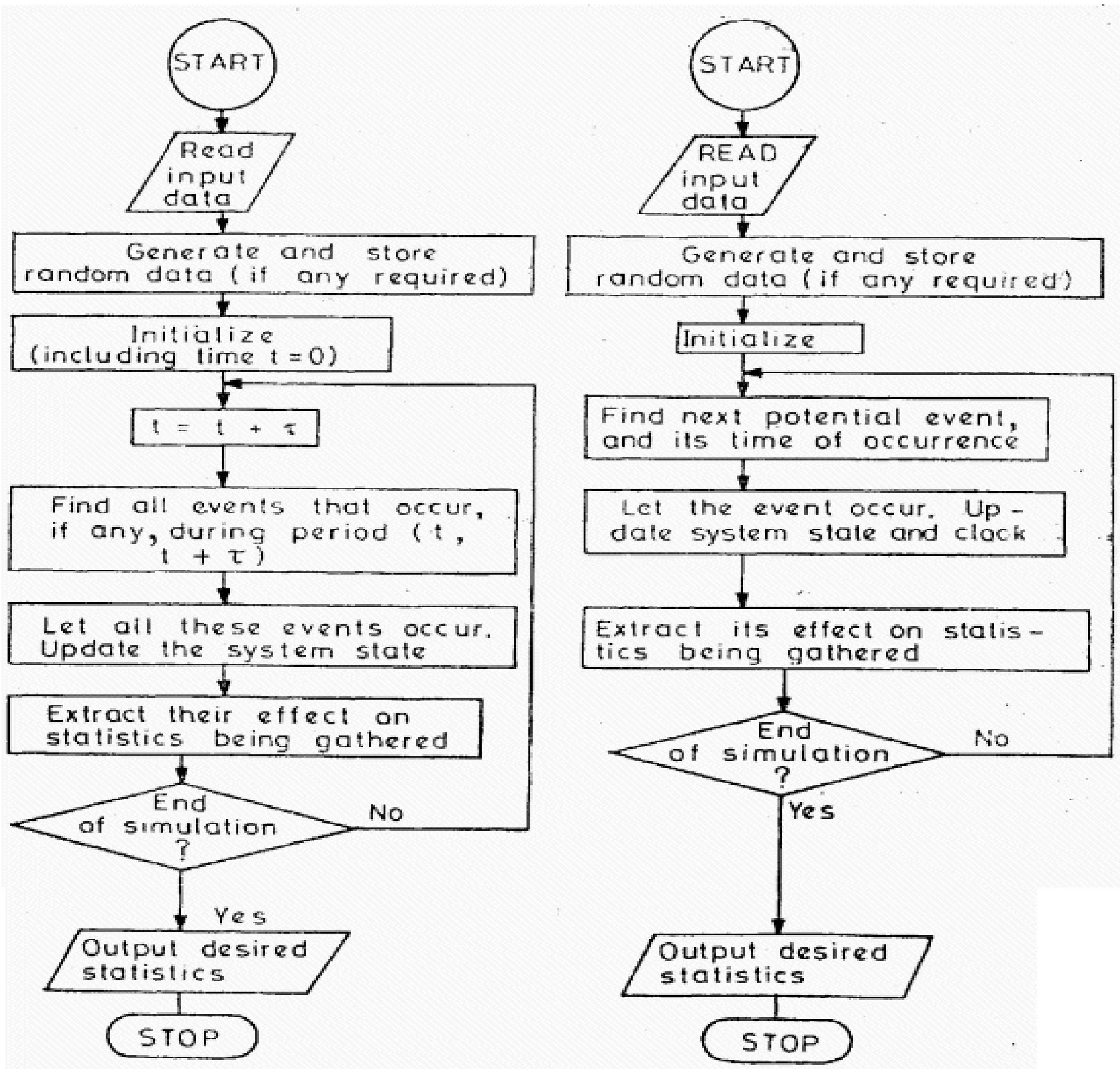
- **System state** – a collection of variables that represent the state of the simulated system.
- **Simulation clock** – a variable that tracks the passage of time in the simulated system.
- **Event list** – a collection of when each type of event will occur.
- **Statistical counters** – variables that contain statistical information about the system, such as average request processing time, server load, or average queue length.
- **Initialization routine** – the routine that defines how the model selects the first event at time 0.
- **Timing routine** – the routine that selects the next event from the event list and jumps in time to its execution.
- **Report generator** – the routine that defines how the program calculates and updates statistical counters and includes them in a post-run report.
- **Event routine** – a routine that updates the system's state when a particular type of event occurs.
- **Library routines** – routines that use probability distributions to generate random values for uncertain variables in the system.
- **Main program** – a program that invokes the model's routines to perform the simulation.

Representation of time in Discrete Event Simulation

Fixed time-step versus next-event model:

In a fixed time-step (**interval Oriented)** model, a timer is simulated by the computer, this timer is updated by a fixed time interval τ, and the system is examined to see if any event has taken place during this time interval, all events that take place during that interval are treated as if they occurred simultaneously at the end of this interval.

In a next-event (**or event-to-event**) model the computer advances time to the occurrence of the next event, thus it shifts from one event to the next; the system state does not change in between. When something of interest happens to the system, the current time is kept track of.
The flowcharts for both models are shown figure below:



(a) Fixed time-step simulation **(b) Next-event simulation**

Comparison between DES and CS:

Aspect	Discrete-Event Simulation (DES)	Continuous Simulation (CS)
Changes in the system	Changes occur only at discrete points in time (events).	The system changes continuously over time.
Primary characteristic	Events trigger changes in the system state.	Time progression drives system behavior.
Time progression	Time jumps from one event to the next.	Time flows continuously without gaps.
State between events	The state remains constant between events.	The state evolves gradually without interruptions.
Use cases	Queue systems, logistics, manufacturing, healthcare.	Physics models, fluid dynamics, weather systems.
Modeling focus	Focuses on discrete entities and events (e.g., arrivals).	Focuses on continuous variables (e.g., velocity, pressure).

Simulation Time and Simulation Clock:

Simulation Time

The simulation time is the logical time within the simulation that represents the point in time currently being simulated. It reflects the system's progression as events are processed.

Simulation Clock

A variable in the simulation program that keeps track of the current simulation time.

Example:

A bank simulation where customers arrive, are served by a teller, and then leave.

Event List:

The following events are scheduled:
- Customer 1 arrives at time 5.
- Customer 1 finishes service at time 10.
- Customer 2 arrives at time 12.

Simulation Process:

Step 1:
- **Simulation Clock:** Initially set to time 0.
- The first event (Customer 1 arrives) is scheduled at time 5.
- The simulation clock advances to time 5 (the time of the first event).
- **Simulation Time** is now 5, representing the system's logical time.

Step 2:

- Process the event at time 5 (Customer 1 arrives).
- The next event (Customer 1 finishes service) is scheduled at time 10.
- Simulation Clock jumps to time 10, skipping idle time.
- Simulation Time is updated to 10.

Step 3:

- Process the event at time 10 (Customer 1 finishes service).
- The next event (Customer 2 arrives) is at time 12.
- Simulation Clock advances to time 12.
- Simulation Time is now 12.

Hence

Simulation Time:

Represents the logical time within the simulation. At any point, it tells you "what time it is" in the simulated system.

Example: At time 5, Customer 1 arrives.

Simulation Clock:

Tracks the current simulation time and controls the system's progression by jumping to the time of the next event.

Example: The simulation clock jumps from time 5 to time 10, skipping the period where nothing happens.

Arrival Processes:

- How customers arrive e.g. singly or in groups (batch or bulk arrivals)
- How the arrivals are distributed in time (e.g. what is the probability distribution of time between successive arrivals (the ***inter-arrival time distribution***)
- Whether there is a finite population of customers or (effectively) an infinite number .

Concept of Arrival Pattern:

In simulation, the concept of "arrival pattern" refers to the pattern or distribution of entities arriving at a particular system or process over time. These entities could be customers, jobs, requests, vehicles, or any other objects that interact with the system being simulated.

Arrivals may occur at scheduled times or at random times. When at random times, the inter arrival times are usually characterized by a probability distribution and most important model for random arrival is the Poisson process. In schedule arrival inter-arrival time of customers are constant.

The arrival pattern is an essential aspect of many simulation models, as it helps to mimic real-world scenarios and analyze system behavior under different conditions.

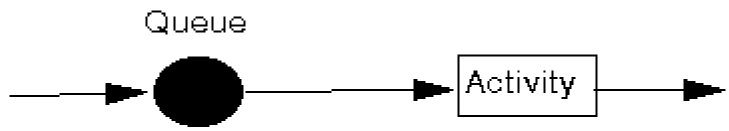


Fig: Simple queuing system

The arrival pattern can be described by various characteristics, including:

- **Arrival Rate:** The rate at which entities arrive per unit of time. It indicates how frequently new entities enter the system. For example, if a simulation is modeling a queue of customers at a store, the arrival rate could be measured in customers per minute.
- **Inter-arrival Time:** The time interval between the arrival of consecutive entities or in another words the inter arrival time is the time between each arrival into the system and the next. It is the reciprocal of the arrival rate.

  inter arrival time = 1/arrival rate

  For example, if the arrival rate is 5 customers per minute, the average inter-arrival time would be 1/5 = 0.2 minutes (or 12 seconds).

  Hence…Inter-arrival time is the average time interval between the arrivals of consecutive entities (e.g., customers, jobs) in a system, and it is calculated as the reciprocal of the arrival rate (Inter-arrival Time=1/Arrival Rate).

- **Arrival Time Distribution:** The probability distribution that describes when entities arrive. Common distributions used in simulation include uniform, exponential, Poisson, normal, and others. Each distribution represents different patterns of arrival times.
- **Arrival Patterns over Time:** The arrival pattern may not remain constant throughout the simulation. In some cases, it might change during different time intervals, reflecting the dynamic nature of real-world systems. For example, in a call center simulation, the arrival rate might increase during peak hours and decrease during off-peak hours.

Generation of arrival pattern using Poisson and Non-stationary Poisson Process

Generating arrival patterns in simulation involves creating a stream of entities arriving at a system according to a specified arrival rate and distribution. The process can be implemented using various techniques, depending on the complexity of the system and the desired characteristics of the arrival pattern.

Some common methods for generating arrival patterns in simulation:

- Stationary Poisson Process
- Non-Stationary Poisson Process
- Batch Arrival

Before explaining the above process lets discuss some terms…

Poisson distribution:

The Poisson distribution is a probability distribution that describes the number of events that occur within a fixed interval of time or space, given a known average rate of occurrence and assuming that events happen independently of each other.

The distribution of the number of arrivals per hour is often modeled using the Poisson distribution. In the context of arrivals per hour, if we assume that arrivals are independent of each other and occur at a constant average rate λ (lambda), then the number of arrivals in any given hour can be modeled using a Poisson distribution with parameter λ.

Note: The Poisson distribution is **discrete**, meaning it deals with countable events (e.g., 0, 1, 2 events). To calculate probabilities for these specific outcomes, we use a PMF.

Poisson probability mass function (PMF):

The Poisson probability mass function (PMF) is used to describe the probability of observing a specific number of events within a fixed interval of time or space, given a known average rate of occurrence (λ) and assuming that events occur independently of each other.

Or in another words, The probability distribution of a discrete random variable is called Probability Mass Function (PMF). It’s a function that maps each value the random variable can take to its corresponding probabilities.

Here's the mathematical expression for the Poisson probability mass function:

$$P(X = k) = \frac{e^{-\lambda} \cdot \lambda^k}{k!}$$

Where:

- P(X=k) is the probability of observing k events in the given interval.

- X: The random variable that follows a Poisson distribution. It represents the number of events that occur within a fixed interval of time or space.
- k: A specific value of the random variable X, indicating the number of events that occurred in the interval.
- e: Euler's number, the base of the natural logarithm (approximately equal to 2.71828).
- λ (lambda): The average rate of occurrence of events in the interval. It's also the parameter of the Poisson distribution.
- k!: The factorial of k, which is the product of all positive integers less than or equal to k.

So, The PMF gives the probability of observing exactly k events in the interval, given the average rate λ.

Poisson Process

Stationary Poisson Process:

A **Stationary Poisson Process** is a special type of Poisson process in which the rate at which events occur is constant over time (i.e., stationary). It is often used to model random events occurring in a fixed interval of time or space, where the probability of an event occurring in any given sub-interval depends only on the length of that interval, not on when the interval occurs.

In this process the *average time* between events is known, but the exact timing of events is random. The average time of event occurrence is denoted by λ (lambda),

Mathematical Representation:

The rate parameter λ represents the expected number of events in one unit of time or space. The number of events X(t) that occur up to time t follows a Poisson distribution with parameter λt:

$$P(X(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

Where:

- X(t) is the number of events in time interval [0,t].
- k is the number of events that occurred in the time interval.
- λ is the rate of occurrence of events per unit time.
- e is Euler's number (approximately 2.71828).

The inter-arrival times (the time between consecutive events) are exponentially distributed with parameter λ, meaning the time between any two consecutive events is exponentially distributed with mean 1/ λ.

Example 1: In a single pump service station, vehicles arrive for fueling with an average of 5 minutes between arrivals.
i.e. 1 car per 5 minutes
If an hour is taken as unit of time, cars arrive according to Poison's process with an average of
λ= 12 cars/hr.
The distribution of the number of arrivals per hour is,

$$f(x) = \Pr(X = x) = \frac{e^{-\lambda}\lambda^x}{x!} = \frac{e^{-12}12^x}{x!}, \begin{cases} x = 0, 1, 2, .. \\ \lambda > 0 \end{cases}$$

Or in other way…….using following formula

$$P(X(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

In this case, we have the following information:

- The average time between arrivals is **5 minutes**, which means the rate of vehicle arrivals λ is 1 vehicle per 5 minutes (i.e., λ=1/5).
- We are interested in calculating the probability that exactly k vehicles arrive in a given time period t.

Let's assume we're considering a 10-minute period, so t=10 minutes.

Given:

- λ=1/5 vehicles per minute
- t=10 minutes
- P(X(t)=k) is the probability of exactly k vehicles arriving in 10 minutes.

Formula:

$$P(X(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

Step-by-Step Calculation:

1. **Determine the value of λt:**

   λt=1/5×10=2

2. **Substitute into the Poisson formula:**

$$P(X(t) = k) = \frac{2^k e^{-2}}{k!}$$

This formula will give us the probability for different values of k (number of vehicles arriving in 10 minutes).

Example Calculation for k=3 (exactly 3 vehicles arriving):

$$P(X(t) = 3) = \frac{2^3 e^{-2}}{3!}$$

Now, let's compute this value.

Calculating:

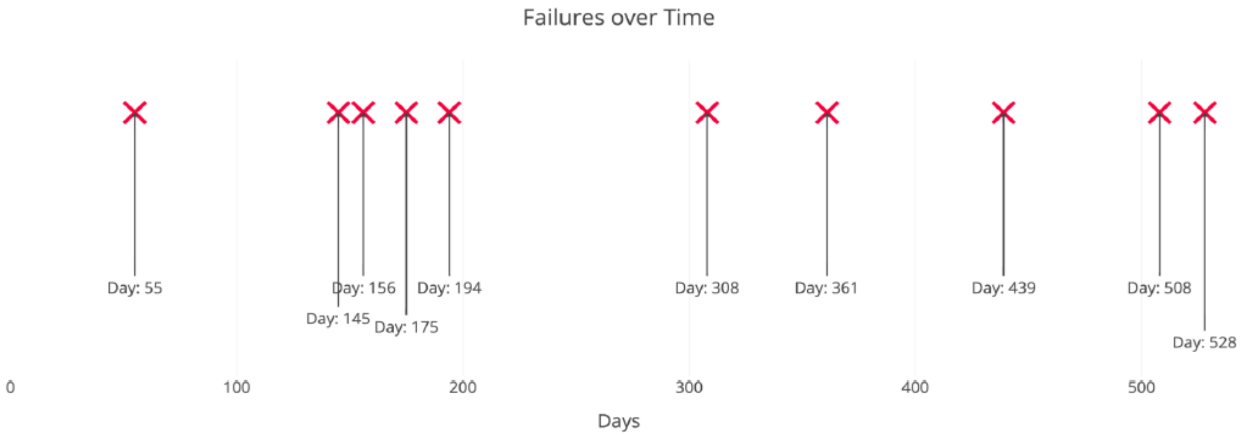
$$P(X(t) = 3) = \frac{8e^{-2}}{6} \approx \frac{8 \times 0.1353}{6} \approx \frac{1.0824}{6} \approx 0.1804$$

So, the probability that exactly 3 vehicles will arrive in the next 10 minutes is approximately **0.1804** or **18.04%**.

Example 2:

Suppose we own a website which our content delivery network (CDN) tells us goes down on average once per 60 days, but one failure doesn't affect the probability of the next. All we know is the average time between failures.

This is a Poisson process that looks like:



Here,

No of event occurs (n) = 10

Total time period (t) = 600

Average time between failure (λ) = 600/10 = 60 days

Rate of occurrence (λ)

In probability/statistics, λ is the rate, not the time between failures.

$\lambda = n/T = 10/600 = 1/60$

$\lambda = 0.0167$ events/day

Average time between failures (Mean inter-arrival time)

This is the reciprocal of λ:

Average time between failures $= 1/\lambda = T/n = 600/10 = 60$ days

Poisson Distribution:

$$P(N(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

Where:

- k = number of events
- t = time period (in days)
- λ = rate of occurrence

Probability of 1 event in 1 day

t=1;

λ=1/60

λt=1/60×1=1/60

$$P(X(t)=k)=\frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

$$P(1)=\frac{(1/60)^1 e^{-1/60}}{1!}$$

$$P(1)\approx 0.0167\times 0.9834$$

$$P(1)\approx 0.0164$$

P(1)≈0.0164

Probability ≈ 0.0164

Probability of exactly k events in 1 day

Here, t=1

$$P(N(t)=k)=\frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

$$P(N(1)=k)=\frac{\left(\frac{1}{60}\right)^k e^{-\frac{1}{60}}}{k!}$$

The important point is we know the *average time between events* but they are randomly spaced (stochastic). We might have back-to-back failures, but we could also go years between failures due to the randomness of the process.

A Poisson Process meets the following criteria (in reality many phenomena modeled as Poisson processes don't meet these exactly):

- Events are **independent** of each other. The occurrence of one event does not affect the probability of another event will occur.
- The average rate (events per time period) is constant.
- Two events cannot occur at the same time.

Common examples of Poisson Process are:

- Customers calling a help center
- Visitors to a website
- Movement in a stock price
- The number of hot dogs sold by Papaya King from 12pm to 4pm on Sunday
- Failures of ultrasound machines in a hospital
- The number of vehicles passing through some intersection from 8am to 11am on weekdays.

<u>Poisson distribution:</u>

The Poisson distribution is a probability distribution that describes the number of events that occur within a fixed interval of time or space, given a known average rate of occurrence and assuming that events happen independently of each other.

The "Events" could be anything like disease cases, customer purchases, failure in system, vehicle movement in road etc. The interval can be any specific amount of time or space, such as 10 days or 5 square inches.

The Poisson distribution has only one parameter, λ (lambda), which is the mean (average) number of events.

A discrete random variable **X** is said to have a Poisson distribution, with parameter **λ > 0,** if it has a probability mass function given by:

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

Where,

- k is the number of occurrences (k=0,1,2,3…..)
- e is Euler's number (e =2.71828)
- λ is average rate of occurrence of variable x

In Poisson distribution, the mean is represented as **E(X) = λ.** For a Poisson Distribution, the mean and the variance are equal. It means that **E(X) = V(X)**, Where, V(X) is the variance.

If percentage of occurrence is given then we can calculate λ by:

λ = np

Where,

n = total number of event

p = percentage of occurrence

You can use a Poisson distribution if:

1. Individual events happen at random and independently. That is, the probability of one event doesn't affect the probability of another event.
2. You know the mean number of events occurring within a given interval of time or space. This number is called λ (lambda), and it is assumed to be constant.

When events follow a Poisson distribution, λ is the only thing you need to know to calculate the probability of an event occurring within a certain number of times.

Example 1:

The number of accident in the street of Kathmandu follows Poisson distribution with mean of 3 accidents per day. Find the probability that

a. No accident occurs in a day
b. More than 3 accident in a day
c. Less than 3 accident in a day
d. Exactly 2 accidents in a day

Solution:

Given, mean (average) accident per day (λ) = 3

a. No accident (k) = 0

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

Where, e = 2.718 (this is constant)

$P(k=0) = (\lambda^0\ e^{-3})/0!$
$= 1*(2.718)^{-3} / 1$
= 0.0498

Hence, probability of no accident occurs in a day is 0.0498.

b. More than 3 accident (k>3)

P(k>3) = 1 – (P(k=0) + P(k=1) + P(k=2) + P(k=3))

New,

P(k=0) = 0.0498
P(k=1) = $(\lambda^1\ e^{-3})/1!$
= (3* 0.0498)/1
= 0.1494

P(k=2) = $(\lambda^2\ e^{-3})/2!$
$= (3^2 * 0.0498)/2$
= (9 * 0.0498)/2
= 0.2241
P(k=3) = $(\lambda^3\ e^{-3})/3!$
$= (3^3 * 0.0498)/6$

= (27*0.0498)/6
= 0.2241
Now
P(k>3) = 1 – (0.0498+0.1494+0.2241 +0.2241)
= 1- 0.6474
= 0.3526
Hence, probability of more than 3 accidents occurs in a day is 0.3526.

c. **Less than 3 accident (k<3)**

P(k<3) = P(k=0) + P(k=1) + P(k=2)
= 0.0498+0.1494+0.2241
= 0.4233
Hence, probability of less than 3 accidents occurs in a day is 0.4233.

d. **Exactly 2 accident in a day (k=2)**

P(k=2) = 0.2241
Hence, probability of exactly 2 accidents occurs in a day is 0.2241.

Example 2:
A manufacturer company of pins knows on an average 2% of its production is defective. Company sells pins in box of 100 and guarantees that not more than 2 pins will be defective in a box. What is the probability that a box selected at random:

a. Will meet the guaranteed quality
b. Will not meet the guaranteed quality

Solution:
Here given,
Average defective percentage (p) = 2% = 0.02
Total no of pins in a box (n) = 100
Average defective pins (λ) = n*p = 100*0.02 = 2
e = 2.718 (this is constant)

a. **Guaranteed will meet when (k<=2)**

P(k<=2) = P(k=0) + P(k=1) + P(k=2)
P(k=0) = $(\lambda^0 e^{-2})/0!$
= (1* 0.1354)/1
= 0.1354
P(k=1) = $(\lambda^1 e^{-2})/1!$
= (2* 0.1354)/1
= 0.2708
P(k=2) = $(\lambda^2 e^{-2})/2!$

= (4* 0.1354)/2
= 0.2708
P(k<=2) = 0.1354+0.2708+0.2708
= 0.677
Hence if box is selected randomly probability of guaranteed quality is 0.677.

b. Guaranteed will not meet when (k>2)

P(k>2) = 1- (P(k=0) + P(k=1) + P(k=2))
= 1- 0.677
= 0.323
Hence if box is selected randomly probability of not guaranteed quality is 0.323.

Example 3:

A random variable X has a Poisson distribution with parameter λ such that
P (X = 1) = (0.2) P (X = 2). Find P (X = 0).

Solution:

For the Poisson distribution, the probability function is defined as:
$P(X = k) = (e^{-\lambda} \lambda^k) / k!$, where λ is a parameter.
Given that, P (X = 1) = (0.2) P (X = 2)
X = k, then
P (k = 1) = (0.2) P (k = 2)
$(e^{-\lambda} \lambda^1) / 1! = (0.2) (e^{-\lambda} \lambda^2) / 2!$
$e^{-\lambda} \lambda = (0.2) (e^{-\lambda} \lambda^2) / 2$
$\lambda = 2/0.2$
$\lambda = 10$
Now, substitute $\lambda = 10$, in the formula, we get:
P (X =0),
X=k then,
$P(k = 0) = (e^{-\lambda} \lambda^0)/0!$
$= e^{-\lambda}$ $(\lambda^0 = 1$ and $0! = 1)$
$= e^{-10}$
= 0.0000454
Thus, probability of zero occurrences is 0.0000454

Example 4:

The average number of goals per match of Messi in the World Cup Soccer is approximately 2.5. Find the probability of 0, 1 or 2 goals of Messi per match in the World Cup.

Solution:

Because the average event rate is 2.5 goals per match, $\lambda = 2.5$.

$$P(k \text{ goals in a match}) = \frac{2.5^k e^{-2.5}}{k!}$$
$$P(k = 0 \text{ goals in a match}) = \frac{2.5^0 e^{-2.5}}{0!} = \frac{e^{-2.5}}{1} \approx 0.082$$
$$P(k = 1 \text{ goal in a match}) = \frac{2.5^1 e^{-2.5}}{1!} = \frac{2.5e^{-2.5}}{1} \approx 0.205$$
$$P(k = 2 \text{ goals in a match}) = \frac{2.5^2 e^{-2.5}}{2!} = \frac{6.25e^{-2.5}}{2} \approx 0.257$$

Non-Stationary Poisson Process (NSPP):

A Non-Stationary Poisson Process (NSPP), also known as an Inhomogeneous Poisson Process, is a generalization of the standard (homogeneous) Poisson process where the arrival rate (or intensity) λ(t) is allowed to vary with time. This contrasts with the standard Poisson process, which assumes a constant arrival rate λ. In other words, λ doesn't change over time this is known as a time-stationary or time-homogenous Poisson process or just simply a stationary Poisson process.

The NSPP is useful for situations in which the arrival rate varies during the period of interest (time), for example, meal times for restaurants, phone calls during business hours, and orders for pizza delivery around 6 P. M. etc.

In NSPP the expected number of arrivals by time t is denoted by Λ(t) also known as cumulative rate function:

Note: The symbol **Λ** is the uppercase Greek letter "Lambda".

$$\Lambda(t) = \int_0^t \lambda(u)\, du$$

Where,

- **Λ(t):** It represents the **cumulative rate function**, also known as the mean value function or the integrated intensity function, which gives the expected number of arrivals or events by time t. It accumulates the rate of occurrences over time.
- **λ(u):** This is the intensity function or **instantaneous rate function**, which represents the rate at which events occur at time u. It can vary over time in a nonhomogeneous Poisson process.
- $\int_0^t \lambda(u)\, du$: This integral calculates the total accumulation of the rate function λ(u) from time 0 to time t, effectively summing up the rate of arrivals over the interval from 0 to t.

Expected Number of Arrivals: The expected number of arrivals by time t is given by the cumulative rate function Λ(t), which is the integral of the rate function λ(t) over time.

$$\mathbb{E}[N(t)] = \Lambda(t) = \int_0^t \lambda(u)\, du$$

Where,

- **E[N(t)]:** This represents the expected number of arrivals by time t. The symbol E stands for the expectation or expected value in probability and statistics.
- **N(t)**: This represents the number of arrivals or events that have occurred by time t in the Non-Stationary Poisson Process (NSPP). It is a random variable because the actual number of arrivals can vary, even though we can calculate an expected value for this number.
- **Λ(t):** This is the cumulative rate function up to time t. It is calculated by integrating the time-varying rate function λ(u) from the start time (0) to the time t.
- **λ(u):** This is the rate function, which describes how the arrival rate of the process changes over time. It is the rate at which events (e.g., helicopter arrivals) occur at any given time u.

Probability of k Arrivals by Time t: The probability that exactly k arrivals occur by time t follows a Poisson distribution with parameter Λ(t).

$$P(N(t) = k) = \frac{(\Lambda(t))^k e^{-\Lambda(t)}}{k!}$$

Let $T_1, T_2, \ldots$ be the arrival times of stationary Poisson process $N(t)$ with $\lambda = 1$, and let $\mathcal{T}_1, \mathcal{T}_2, \ldots$ be the arrival times for an NSPP $\mathcal{N}(t)$ with arrival rate $\lambda(t)$. The fundamental relationship for working with NSPPs is the following:

$$T_i = \Lambda(\mathcal{T}_i)$$
$$\mathcal{T}_i = \Lambda^{-1}(T_i)$$

In other words, an NSPP can be transformed into a stationary Poisson process with arrival rate 1, and a stationary Poisson process with arrival rate 1 can be transformed into an NSPP with rate $\lambda(t)$, and the transformation in both cases is related to $\Lambda(t)$.

Example 1:

- Suppose we conducting a time study of a helicopter maintenance facility
- Our data indicates that the facility is busier in the morning than the afternoon
  - In the morning (0900-1300): expected inter-arrival time of 0.5 hours
  - In the afternoon(1300-1700): expected inter-arrival time of 2 hours

a. What is the probability that 2 helicopters arrive between 1200 and 1400 , given that 5 arrived between 0900 and 1200?
b. What is the expected number of helicopters to arrive between 1200 and 1400?

Solution:

since $\lambda = \frac{1}{\text{inter-arrival time}}$

Let's say

t = 0 correspond to 0900
t = 3 correspond to 1200
t = 5 correspond to 1400
t = 8 correspond to 1700

Therefore, the arrival rate $\lambda(t)$ as a function of t (in hour) is:

$\lambda(t) = 1/0.5= 2$ if $0 \le t < 4$

$\lambda(t) = 1/2= 1/2$ if $4 \le t < 8$

Using this we can compute the integrated-rate function $\Lambda(t)$, or the expected number of arrivals by time t:

Expected number of arrivals = 2.5 helicopters

$$\Lambda(t) = \int_0^t \lambda(s)\,ds$$

If $t \in [0,4]$:

$$\Lambda(t) = \int_0^t 2\,ds$$

$$\Lambda(t) = 2[s]_0^t$$
$$= 2[t-0]$$
$$= 2t$$

Similarly

If $t \in [4,8]$:

$$\Lambda(t) = \int_0^4 2\,ds + \int_4^t \frac{1}{2}\,ds$$
$$= 2[s]_0^4 + \frac{1}{2}[s]_4^t$$
$$= 2[4-0] + \frac{1}{2}(t-4)$$
$$= 8 + \frac{t}{2} - 2$$
$$= \frac{t}{2} + 6$$

$\therefore$

$\Lambda(t) = 2t$ if $0 \le t < 4$

$\Lambda(t) = \frac{t}{2} + 6$ if $4 \le t < 8$

a. What is the probability that 2 helicopters arrive between 1200 and 1400, given that 5 arrived between 0900 and 1200?

(a) given

$P_r[N(3) - N(0)] = 5]$

$P_r[N(5) - N(3) = 2]$

Now,

$\Lambda(t) = N[5] - N(3)$

$= \Lambda(5) - \Lambda(3)$

$= \frac{t}{2} + 6 - 2t$

$= \frac{5}{2} + 6 - 2 \times 3$

$= \frac{17}{2} - 6 = \frac{17 - 12}{2}$

$= \frac{5}{2}$

$= 2.5$

$\therefore \quad P(K=2) = \frac{e^{-\Lambda(t)} \Lambda(t)^K}{K!}$

$= \frac{e^{-2.5} \times (5/2)^2}{2!} = \frac{(2.718)^{-2.5} \times (2.5)^2}{2}$

$= \frac{0.5125}{2} = 0.26$

Hence, the probability of 2 helicopter arrive between 1200 to 1400 is 0.26

b. What is the expected number of helicopters to arrive between 1200 and 1400?

(b) $E[N(5) - N(3)]$

$= \Lambda(5) - \Lambda(3)$

$= \frac{t}{2} + 6 - 2t$

$= \frac{5}{2} + 6 - 2 \times 3$

$= \frac{5}{2}$

$= 2.5$

Hence, expected number of helicopter will arrive between 1200 to 1400 is 2.5

Solved in another way:

Question:

- Suppose we conducting a time study of a helicopter maintenance facility
- Our data indicates that the facility is busier in the morning than the afternoon
  - ➢ In the morning (0900-1300): expected inter-arrival time of 0.5 hours
  - ➢ In the afternoon(1300-1700): expected inter-arrival time of 2 hours

c. What is the probability that 2 helicopters arrive between 1200 and 1400 , given that 5 arrived between 0900 and 1200?

d. What is the expected number of helicopters to arrive between 1200 and 1400?

Solution:

Given that the inter-arrival times and arrival rates vary between the morning and afternoon, we need to model this situation using a nonstationary Poisson process.

- **Morning (0900-1300)**: Expected inter-arrival time is 0.5 hours, so the arrival rate
  $\lambda_{morning}$=1/ inter-arrival time
  =1/0.5
  = 2 helicopters per hour.

- **Afternoon (1300-1700)**: Expected inter-arrival time is 2 hours, so the arrival rate
  $\lambda_{afternoon}$=1/ inter-arrival time
  =1/2
  = 0.5 helicopters per hour.

For the nonstationary Poisson process, the arrival rate changes over time. We need to calculate probabilities and expectations for the interval from 1200 to 1400, which includes both the end of the morning period and the start of the afternoon period.

Part (a): Probability that 2 helicopters arrive between 1200 and 1400, given that 5 arrived between 0900 and 1200

Given that we already know 5 helicopters arrived between 0900 and 1200, we focus on the interval between 1200 and 1400. This interval is divided into two parts:

- **From 1200 to 1300:**
  Arrival rate $\lambda_{1200-1300}$= 2 helicopters per hour.

- **From 1300 to 1400:**
  Arrival rate $\lambda_{1300-1400}$ = 0.5 helicopters per hour.

For a nonstationary Poisson process, we can sum the expected number of arrivals Λ(t) for these two intervals.

Calculating the expected number of arrivals

To calculate the expected number of arrivals in a non-stationary Poisson process, we use the formula for the expected arrival rate Λ(t) over a time interval t, which is given by the integral of the rate function λ(s) over that interval that is …

$$\Lambda(t) = \int_0^t \lambda(s)\, ds$$

- **From 1200 to 1300:**

$$\Lambda_{1200-1300} = \int_{1200}^{1300} 2\, ds = 2 \times (1300 - 1200) = 2 \times 1 = 2$$

- From 1300 to 1400:

$$\Lambda_{1300-1400} = \int_{1300}^{1400} 0.5\, ds = 0.5 \times (1400 - 1300) = 0.5 \times 1 = 0.5$$

So, the total expected number of arrivals from 1200 to 1400 is:

$$\Lambda_{1200-1400} = \Lambda_{1200-1300} + \Lambda_{1300-1400} = 2 + 0.5 = 2.5$$

Hence,

$$\Lambda(1200, 1400) = 2.5$$

Now, Probability that 2 helicopters arrive between 1200 and 1400

For the nonstationary Poisson process, the number of arrivals in the interval [1200, 1400] follows expected number of arrivals $\Lambda(t)$.

The probability of observing k arrivals in a Poisson process with expected number of arrivals $\Lambda(t)$ is given by:

$$P(X = k) = \frac{e^{-\Lambda(t)}\Lambda(t)^k}{k!}$$

Here ,

$\Lambda(t)$ =2.5 and k=2

$$P(X = 2) = \frac{e^{-2.5} \cdot 2.5^2}{2!} = \frac{e^{-2.5} \cdot 6.25}{2} = 6.25 \cdot \frac{e^{-2.5}}{2}$$

Using $e^{-2.5} \approx 0.0821$:

$$P(X = 2) = 6.25 \cdot \frac{0.0821}{2} = 6.25 \cdot 0.04105 \approx 0.2566$$

So, the probability that 2 helicopters arrive between 1200 and 1400 is approximately 0.2566.

Part (b): Expected number of helicopters to arrive between 1200 and 1400

The expected number of helicopters arriving between 1200 and 1400 is the sum of the expected arrivals for each sub-interval (1200-1300 and 1300-1400):

$E[X_{1200-1300}] = \Lambda(t)_{1200-1300} = 2$
$E[X_{1300-1400}] = \Lambda(t)_{1300-1400} = 0.5$

Thus, the total expected number of helicopters arriving between 1200 and 1400 is:
$E[X_{1200-1400}] = E[X_{1200-1300}] + E[X_{1300-1400}]$
$= 2 + 0.5$
$= 2.5$

Therefore, the expected number of helicopters to arrive between 1200 and 1400 is 2.5.

Batch/Bulk Arrival:

In ordinary arrivals, customers come one by one, so arrival times are strictly increasing:

$t0<t1<t2<\cdots t$

But in batch (bulk) arrivals, several customers arrive at exactly the same time (for example, a bus unloads passengers).

So we allow equal arrival times:

$t0\le t1\le t2\le\cdots t$

Each tn represents the arrival time of the n-th customer, not the n-th batch.

If arriving customers to a queue occur in "batches" such as busloads, then we can model this by a point process $\psi = \{t_n\}$ in which the arrival times of customers can coincide: $t0 \le t1 \le t2 \le \cdot\cdot\cdot$, where $\lim_{n\to\infty} t_n = \infty$. Since the limit is infinite, we conclude that the inequalities must consist of an infinite number of strict inequalities with a finite number of equalities in between.

Hence…

If multiple customers arrive together, they all get the same arrival time.

- Equal times → customers in the same batch
- A jump to a larger time → next batch

For example: $0 = t0 = t1 = t2 < 1 = t3 = t4 = t5 = t6 < 3 = t7 = t8 < t9 \cdot\cdot\cdot$

This means…

(a) At time t=0

$0=t_0=t_1=t_2$

- Three arrival times are equal
- So 3 customers arrive at time 0

Batch size = 3 at t=0

(b) At time t=1

$1=t3=t4=t5=t6$

- Four arrival times are equal
- So 4 customers arrive together at time 1

Batch size = 4 at t=1

(c) At time t=3

$3=t7=t8$

- Two arrival times are equal
- So 2 customers arrive together at time 3

Batch size = 2 at t=3

(d) After that

t9<⋯

- The next customer arrives later
- Could be a single arrival or another batch

If we randomly select an integer j, then Cj (the jth customer; arrival time tj) is a member within some batch. As is underlying the so called inspection paradox, we are more likely to choose someone from a larger batch since larger batches contain more customers. The size of this batch is thus biased to be larger than usual.

Note: The symbol **ψ** is the lowercase Greek letter "Psi," pronounced as "sigh" or "psigh."

Gathering Statistics:

Most simulation programming system includes a report generator to print out statistics gathered during the run. The exact statistics required from a model depend upon the study being performed, but there are certain commonly required statistics which are usually included in the output. Among those commonly needed statistics are:

1. ***Counts:*** Giving the number of entities of a particular type or the number of times some event occurred.
2. ***Summary measures:*** Such as extreme values, mean values, and standard deviations.
3. ***Utilization:*** Defined as the fraction (or percentage) of time some entity is engaged.
4. ***Occupancy:*** Defined as the fraction (or percentage) of a group of entities in use on the average.
5. ***Transit times:*** defined as the time taken for an entity to move from one part to the system to some other part.
6. Distributions of important variables such as queue length or waiting times.

When there are stochastic effects operating in the system, all these system measures will fluctuate as a simulation proceeds, and the particular values reached at the end of the simulation are taken as estimates of the true values they are designed to measure.

Monte Carlo Method Simulation:

A Monte Carlo simulation is a computational technique that uses random sampling to model and analyze complex systems or processes.

In a Monte Carlo simulation, you start with a mathematical model representing the system or process you want to study. This model contains various input variables or parameters varying within certain ranges. By randomly sampling the values for these variables, you can generate many simulations or iterations, each representing a possible system outcome.

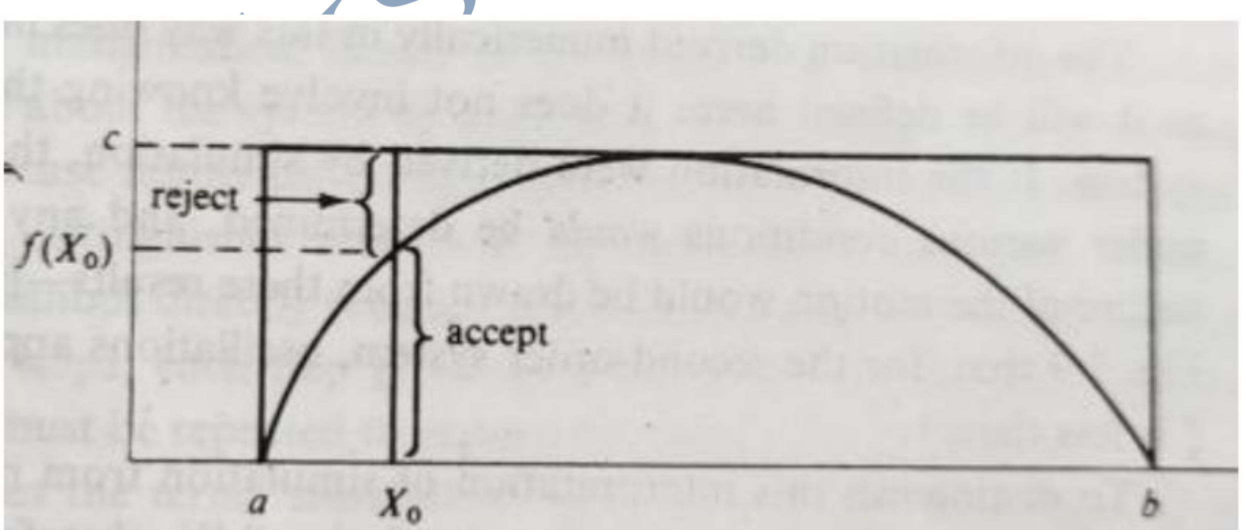
For each iteration, the model is run using the sampled values, and the resulting outputs or outcomes are recorded. These outputs could be numerical values, statistical measures, or other relevant information based on the system being simulated. By collecting data from multiple iterations, you can analyze the distribution of outcomes and conclude the behavior and characteristics of the system.

Monte Carlo simulations are particularly useful when dealing with complex systems with high uncertainty or randomness. They are widely applied in various fields, such as finance, engineering, physics, economics, and risk analysis, among others.

A Monte Carlo simulation takes the variable that has uncertainty and assigns it a random value. The model is then run and a result is provided. This process is repeated again and again while assigning many different values to the variable in question. Once the simulation is complete, the results are averaged to arrive at an estimate.

Example:

Suppose the function, ***f(x)*** is positive and has lower and upper bounds ***a*** and ***b***, respectively. Suppose, also, the function is bounded above by the value ***c***.



As shown in the figure above, the graph of the function is then contained within a rectangle with sides of length ***c***, and ***b-a*.** If we pick points at random within the rectangle, and determine whether they lie beneath the curve or not, it is apparent that, providing the distribution of selected points is uniformly spread over the rectangle. The fraction of the points falling on or below the curve should be approximately the ratio of the area under the curve to the area of the rectangle. If ***N*** points are used and ***n*** of them fall under the curve, then, approximately,

$$\frac{n}{N} = \int_a^b \frac{f(x)\,dx}{c(b-a)}$$

The accuracy improves as the number of ***N*** increases. When it is decided that the sufficient points has been taken, the value of integral is estimated by multiplying ***n/N*** with the area of rectangle, ***c(b-a).***

Example: Estimating the value of Pi using Monte Carlo Simulation

Let's consider a simple mathematical example of estimating the value of π (pi) using a Monte Carlo simulation. In this case, we can use it to approximate the value of π.

Here is the basic idea:

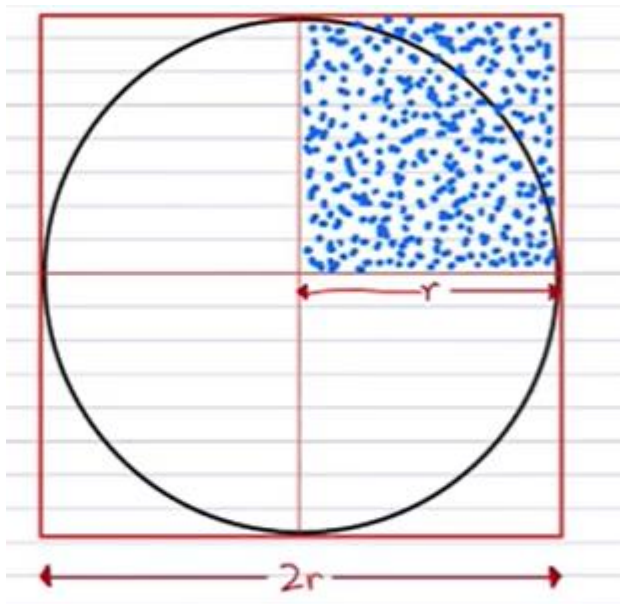
1. Consider a square with a side length of 2 units, centered at the origin (0, 0)
2. Inside the square, inscribe a circle with a radius of 1 unit, also centered at the origin
3. The area of the square is $l^2 = l*l = 2\times2 = 4$ square units,
   while the area of the circle is= $\pi r^2 = \pi \times 1^2 = \pi$ square units

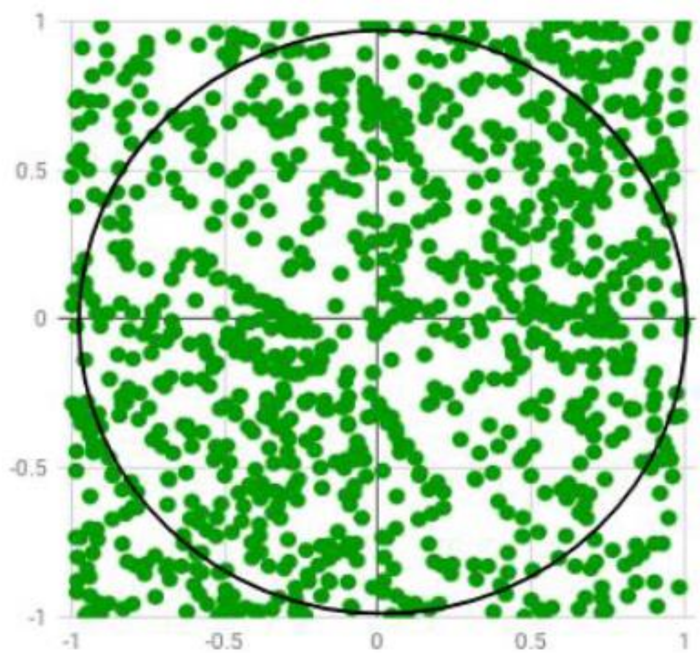
4. The ratio of the areas of the circle to the square is…..

   area of the circle = πr^2
   area of the square = $(2r)^2$
   Ratio= area of the circle/ area of the square = $\pi r^2/(2r)^2 = \pi/4$

   Now if we multiply both side by 4 we get
   Ratio x 4= $(\pi/4) \times 4$
   That is …

Ratio x 4= π
Or
π = Ratio x 4





To estimate the value of π using a Monte Carlo simulation, we can follow these steps:

1. Generate a large number of random points (x, y) within the square
2. Count the number of points that fall within the circle (i.e., satisfy the equation x^2 + y^2 ≤ 1)
3. Calculate the ratio of the number of points inside the circle to the total number of points generated (Area of circle to area of square)
4. Multiply the ratio by 4 to estimate the value of π.

Let's say we generated N random points. If M of those points falls inside the circle, the estimated value of π can be calculated as:

π ≈ 4 * (M / N)

As the number of random points increases, the estimate becomes more accurate. This is because the ratio of the areas of the circle to the square approaches the actual value of π/4.

By repeating this simulation several times with different sets of random points, we can obtain a range of estimates for π. Taking the average of these estimates will yield a more precise approximation of the value of π.

That's a simple example of how a Monte Carlo simulation can be used to estimate the value of π using random sampling.

Describing in another way

The idea is to simulate random points (x, y) in a 2-D plane with domain as a square of side 1 unit. Imagine a circle inside the same domain with same diameter and inscribed into the square. We then calculate the ratio of number points that lied inside the circle and total number of generated points.

Let, (x_0,y_0) be an initial guess for the sample point then form linear congruential method of random number generation:

$x_{i+1} = (ax_i + c) \bmod m$

$y_{i+1} = (ay_i + c) \bmod m$

Where, a and c are constant, m is the upper limit of generated random number

Distance from center (x_0,y_0) to any point (x_n,y_n) is

$d = x^2 + y^2$

We know that,

Area of circle = πr^2

Area of quadrant of circle = $\frac{1}{4}\pi r^2$

Area of square = r^2

Now,

$$\frac{\text{Area of quadreant of circle}}{\text{Area of square}} = \frac{\text{Number of points inside the circle}}{\text{Total numbers of points generated}}$$

$$\frac{\frac{1}{4}\pi r2}{r2} = \frac{n}{N}$$

$$\pi = 4\frac{n}{N}$$

The Algorithm

1. Initialize circle_points, square_points and interval to 0.
2. Generate random point x.
3. Generate random point y.
4. Calculate d = x*x + y*y.
5. If d <= 1, increment circle_points.
6. Increment square_points.
7. Increment interval.
8. If increment < NO_OF_ITERATIONS, repeat from 2.
9. Calculate pi = 4*(circle_points/square_points).
10. Terminate.

Practical -1: Estimating the value of Pi using Monte Carlo Simulation

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
double monte_carlo_pi_estimation(int no_of_iterations)
  {
          int circle_points = 0;
          int square_points = 0;
          int interval = 0;

          // Seed the random number generator
         srand(time(NULL));
         while (interval < no_of_iterations)
             {
                 // Generate random point (x, y)
                  double x = (double)rand() / RAND_MAX * 2.0 - 1.0;
                  double y = (double)rand() / RAND_MAX * 2.0 - 1.0;


                 // Calculate d = x*x + y*y
                    double d = x * x + y * y;
```

```
                    // Check if the point is inside the circle
                        if (d <= 1)
                            {
                              circle_points++;
                            }

                    // Increment the total number of points
                            square_points++;
                             interval++;
                }


        // Calculate the estimated value of Pi double

                    pi_estimate = 4.0 * ((double)circle_points / (double)square_points);
                    return pi_estimate;
}




int main()
   {

              int no_of_iterations = 1000000;

             // Number of iterations double
              pi_estimate = monte_carlo_pi_estimation(no_of_iterations);
              printf("Estimated value of Pi with %d iterations: %f\n",
                         no_of_iterations, pi_estimate);

               return 0;
   }
```

End of Unit-2