**INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR**

****

**RELIABILITY SIMULATION LABORATORY**

**RE69004**

**Subir Chowdhury School of Quality and Reliability**

**Spring (2025 – 2026)**

**By**

**SUTAMA SARKAR (25RE61R07)**

**ACKNOWLEDGEMENT**

1. I would like to express my deepest gratitude to Prof Sanjay Kumar Chaturvedi and Prof. Indranil Hazra whose expert guidance and insightful teachings in Reliability Simulation Lab have been invaluable throughout this laboratory work. Their continuous encouragement and rigorous academic inputs have significantly enhanced my understanding and skills on the subject. I am truly thankful for his mentorship and the time he devoted to helping us strengthen our analytical and practical abilities in this domain.

2. I extend my heartfelt gratitude to Mr. Tanmay Mondal, our lab-in-charge, for his consistent assistance, smooth coordination of lab activities, and technical support.

3. Finally, I thank all my seniors and fellow students for their cooperation, collaborative learning spirit, and mutual support throughout the lab work, which made this journey both productive and enriching.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **NO** | **TOPIC** | **PAGE NUMBER** |
| **1** | **Assignment 1(Coin,Dice,Pi value)** | **4-19(a)** |
| **2** | **Assignment 2** | **1-20(b)** |
| **3** | **Assignment 3** | **21-30** |
| **4** | **Capacity Demand problem** | **31-36** |
| **5** | **MTTF problem** | **37-40** |
| **6** | **Integration problem** | **41-43** |
|  |  |  |

**RSL: ASSIGNMENT 1**

**PROBLEM\_1: *Simulation of coin toss experiment***

(a) Simulate a fair coin toss experiment for 𝑁 = 100,1000,10000 trials.

(b) Estimate the probability of obtaining heads and plot its convergence with increasing 𝑁.

(c) Modify the simulation to represent a biased coin with probability of heads 𝑝 = 0.6.

(d) Repeat the simulation and compare the estimated probability with the true value.

(e) Compute and plot the absolute estimation error as a function of the number of trials.

* 1. **METHODOLOGY**

*Analytical approach:* Each coin toss is modelled as a Bernoulli random experiment with two possible outcomes:

* Head (success) → X=1
* Tail (failure) → X=0

Let:

P(X=1) = p

where p is the true probability of obtaining heads.

The estimated probability after N trials is:

According to the Law of Large Numbers:

This implies that as the number of trials increases, the estimated probability approaches the true probability of heads.

The absolute estimation error is defined as:

Error =

For Monte Carlo simulations, the error decreases approximately as:

*Simulation approach***:** The simulation employs a Monte Carlo approach using uniformly distributed random numbers to model Bernoulli trials and estimate coin toss probabilities through repeated random sampling.

**1. Generate uniform random variables**

**2. Convert them into coin toss outcomes:** For a fair coin, the probability of head in a fair coin is 0.5 and according to the problem statement the probability of head in a biased coin 0.6

For fair coin: If Ui ≤ 0.5 → Xi = 1 (Head)  
 If Ui > 0.5 → Xi = 0 (Tail)

For a biased coin: If Ui ≤ 0.6 → Yi = 1 (Head)  
 If Ui > 0.6 → Yi = 0 (Tail)

**3. Repeat for N trials**

The total number of heads after N trials is:

Headcount = X1 + X2 + ... + XN (fair coin)

Or

Headcount = Y1 + Y2 + ... + YN (biased coin)

**4. Estimate probabilities**

P(H) = Headcount / N

**5. Analyse convergence and error**

Error = | P(H) − p |

**1.2: Flowchart and Algorithm:**

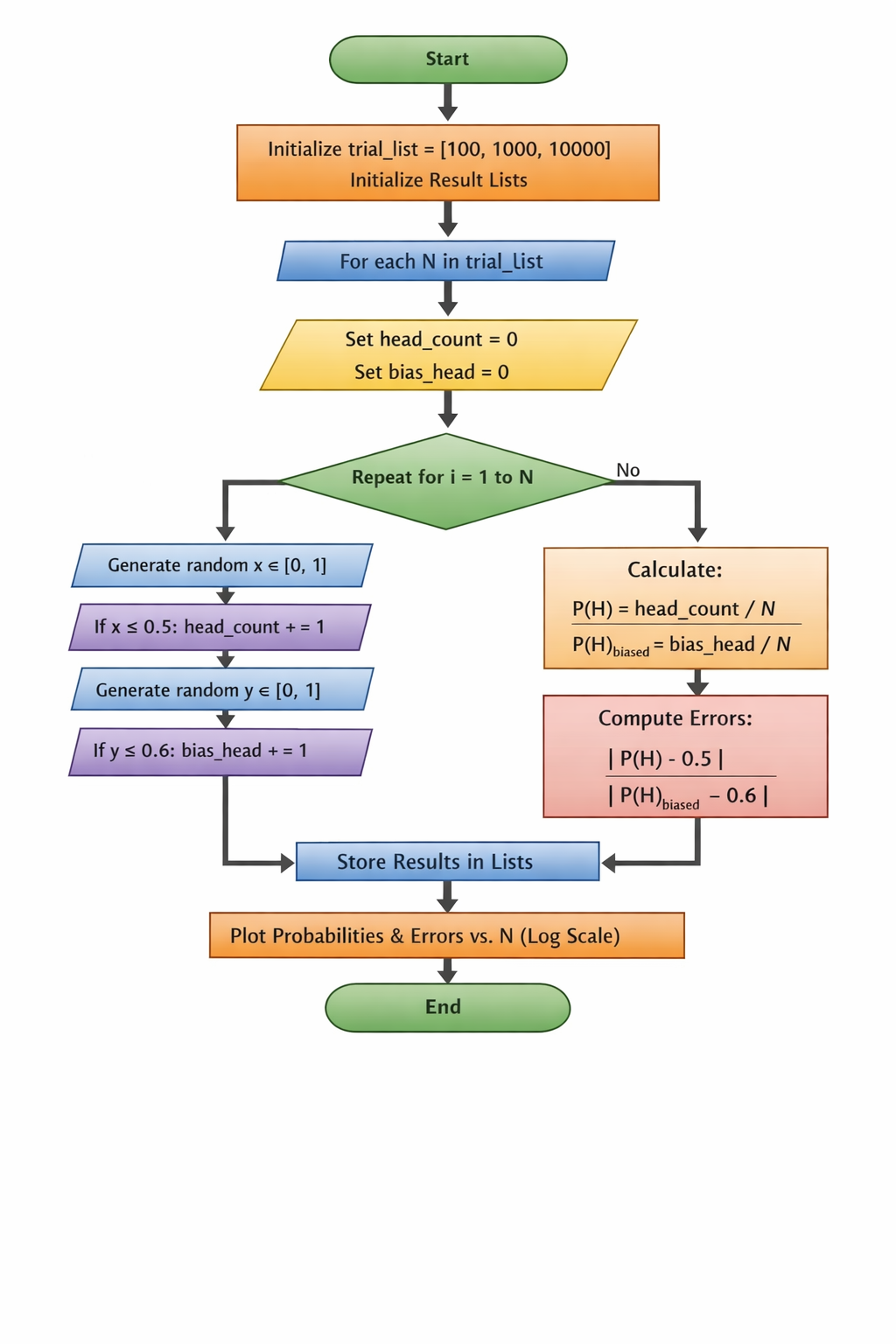
*Flowchart:*

Fig 1.2.1: Flowchart of coin toss problem (simulation approach)

# *Algorithm*:

**Step 1:** Start

**Step 2:** Initialize the list of trials: trial\_list = [100, 1000, 10000]

**Step 3:** Initialize empty lists to store results  
**Step 4:** For each value of N in trial\_list, perform the following steps:

**Step 4.1:** Set head\_count = 0 and bias\_head = 0

**Step 4.2:** Repeat for i = 1 to N:  
       Generate a random number x in [0,1]  
       If x ≤ 0.5, increment head\_count

Generate another random number y in [0,1]  
       If y ≤ 0.6, increment bias\_head

**Step 5:** Compute fair coin probability:  P\_fair = head\_count / N

**Step 6:** Compute biased coin probability: P\_biased = bias\_head / N

**Step 7:** Compute absolute errors: Error\_fair = |P\_fair − 0.5|  
       Error\_biased = |P\_biased − 0.6|

**Step 8:** Store all values in corresponding lists

**Step 9:** Display the results in a formatted table

**Step 10:** Plot probabilities and absolute errors versus number of trials on a logarithmic scale

**Step 11:** Stop

**1.3:Result and Outcome:**

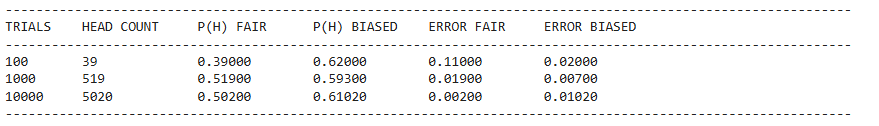


Fig 1.3.1: Table of the simulation experiment

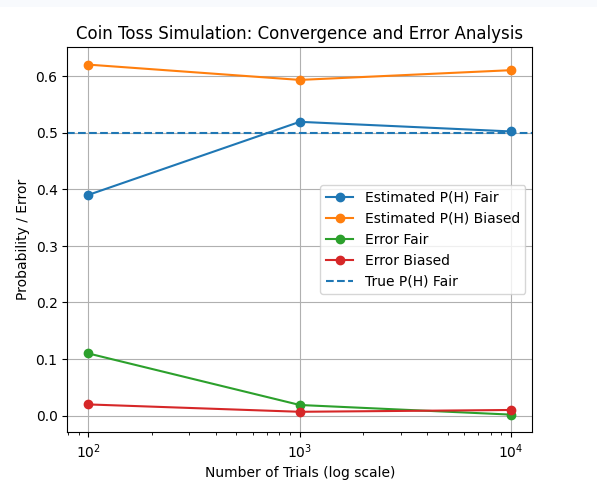


Fig 1.3.2: Graphical representation of Probability/Error vs Number of Trials (log scale)

## **1.4. Observations**

• Small sample sizes produce high fluctuations and larger error  
• Increasing trials improves estimation accuracy  
• Both fair and biased probabilities converge to true values  
• Error decreases approximately with increasing N  
• Monte Carlo simulation effectively validates theoretical probability laws

## **1.5. Discussion**

The estimated probabilities for both fair and biased coins move closer to their true values as the number of trials increases. Large deviations are observed for small sample sizes due to randomness. The absolute estimation error decreases with increasing trials, showing improved accuracy. This behaviour confirms convergence predicted by the Law of Large Numbers.

## **1.6. Conclusion**

The simulation confirms that probability estimates converge to their true values as the number of trials increases, while absolute estimation error decreases, demonstrating the Law of Large Numbers and the effectiveness of Monte Carlo methods.

**Problem\_2:** *Dice roll simulation and probability estimation*

(a) Simulate the rolling of a fair six-sided die for 𝑁 = 1000, 10000, 100000 trials.

(b) Estimate the probability of each face and verify uniformity.

(c) Estimate the probability of the following events: (i) Rolling an even number

(ii) Rolling a number greater than 4

**2.1. METHODOLOGY**

*Analytical Approach:* A fair six-sided die is modelled as a discrete uniform random variable X taking values from 1 to 6. Each outcome has equal probability:

P(X = k) = 1/6=0.1667 , for k = 1, 2, 3, 4, 5, 6

Theoretical probabilities of composite events are obtained using basic probability rules:

Probability of an even number:

P(Even) = P(2) + P(4) + P(6) = 3/6 = 0.5

Probability of a number greater than four:

P(X > 4) = P(5) + P(6) = 2/6 = 1/3=0.33

These analytical values serve as references to evaluate the simulation results.

*Simulation Approach:* The probability experiment is simulated using the Monte Carlo random sampling technique. A fair six-sided die is modeled by generating random integers between 1 and 6 for each trial.

Let Xi represent the outcome of the i-th trial:

Xi ∈ {1, 2, 3, 4, 5, 6}

Each value occurs with equal likelihood.

For a given number of trials N, the simulation is repeated N times and the frequency of each outcome is counted:

Count(k) = number of times outcome k occurs, where k = 1, 2, 3, 4, 5, 6

The estimated probability of each outcome is computed as:

P̂(k) = Count(k) / N

Composite event probabilities are obtained as:

Probability of even number:

P̂(Even) = [Count (2) + Count (4) + Count (6)] / N

Probability of number greater than four:

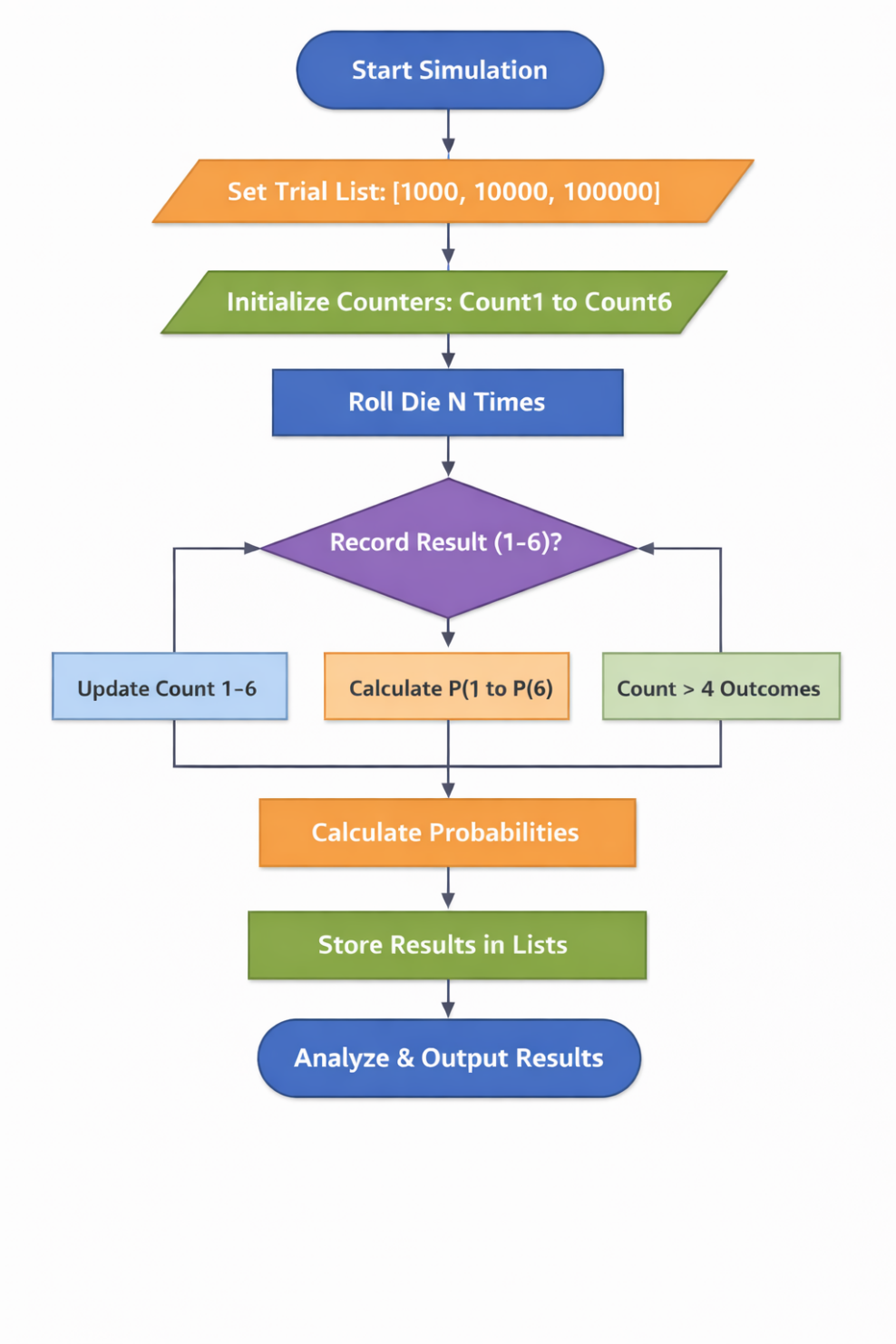
P̂ (Greater than four) = [Count (5) + Count (6)] / N

The experiment is performed for increasing values of N ranging from 1000 to 10,000,0 to observe convergence.

The estimated probabilities are plotted against the number of trials using a logarithmic scale to analyse convergence behaviour.

As N increases, the estimated probabilities approach their theoretical values, demonstrating the Law of Large Numbers.

**2.2. Flowchart and algorithm**

****

Count even number

Fig 2.2.1: Flowchart of dice simulation experiment

# *Algorithm:*

**Step 1:** Start

**Step 2:** Initialize trial values:  
N = [1000, 10000, 100000]

**Step 3:** Initialize empty lists to store probabilities and errors

**Step 4:** For each value of N, perform the following:

**Step 4.1:** Set counters for each face (1 to 6) equal to zero

**Step 4.2:** Repeat N times:  
      Generate a random integer between 1 and 6  
      Increment the corresponding face counter

**Step 4.3:** Compute estimated probabilities:

P̂(1) = C1 / N  
P̂(2) = C2 / N  
P̂(3) = C3 / N  
P̂(4) = C4 / N  
P̂(5) = C5 / N  
P̂(6) = C6 / N

**Step 4.4:** Compute composite event probabilities:

P̂(EVEN) = (C2 + C4 + C6) / N  
P̂(>4) = (C5 + C6) / N

**Step 4.5:** Compute absolute errors:

Error(EVEN) = | P̂(EVEN) − 0.5 |  
Error(>4) = | P̂(>4) − 2/6 |

**Step 4.6:** Store all probabilities and errors

**Step 5:** Plot probability estimates versus number of trials

**Step 6:** Plot absolute errors versus number of trials

**Step 7:** Stop

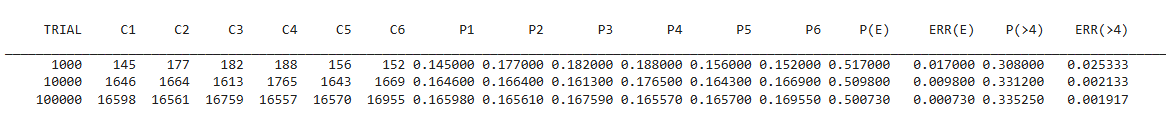
**2.3. Result and Outcome**:

Fig 2.2.2: Table of the simulation experiment

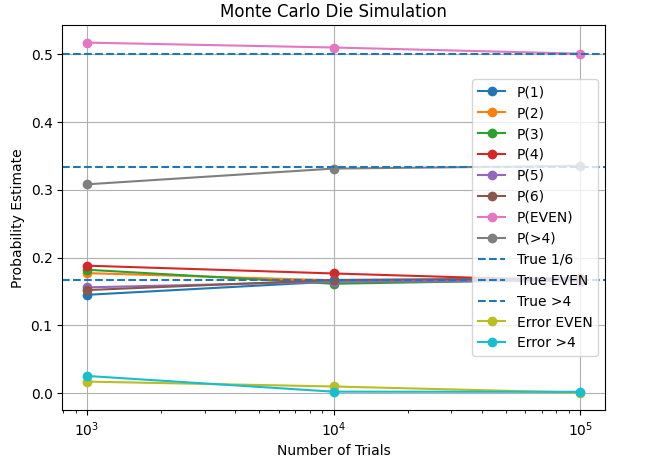


Fig 2.2.3: Graphical representation of Probability Estimate vs Number of Trials

**2.4.Observations**

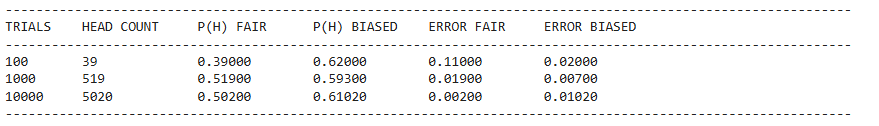
• The estimated probabilities of all six die faces gradually approach the theoretical value of 1/6 as the number of trials increases.  
• The probability of obtaining an even number converges toward 0.5.  
• The probability of obtaining a number greater than four converges toward 1/3.  
• The absolute errors for both composite events decrease significantly with increasing trials

## **2.5. Discussion**

For small sample sizes, noticeable fluctuations occur in the estimated probabilities due to randomness. As the number of trials increases from 1,000 to 100,000, these fluctuations reduce and the estimates stabilize near their theoretical values. The decreasing absolute error confirms improved accuracy and validates the Law of Large Numbers. The Monte Carlo simulation effectively models the random behaviour of die rolls.

## **2.6. Conclusion**

The Monte Carlo simulation successfully estimates the probabilities of dice outcomes and composite events. Increasing the number of trials improves accuracy and reduces estimation error. The results confirm theoretical probability values and demonstrate convergence behaviour predicted by the Law of Large Numbers.

Bottom of Form

**Problem\_3:** *Estimation of π using Monte Carlo simulation*

(a) Use Monte Carlo simulation to estimate the value of π by randomly generating points inside a unit square.

(b) Determine whether a point lies inside the largest circle that can be inscribed in the square.

(c) Estimate π using

≈ 4× (Number of points inside the circle /Total number of points)

(d) Perform simulations for increasing sample sizes 𝑁 = 1000,10000,100000

(e) Plot the estimated value of π and the estimation error as a function of 𝑁.

**3.1. METHODOLOGY**:

*Analytical Approach:* A square of side length 1 is considered with a circle of radius 0.5 inscribed inside it.

Area of the square = 1 × 1 = 1

Area of the circle = π × (0.5) ² = π/4

The ratio of the circle area to the square area is therefore:

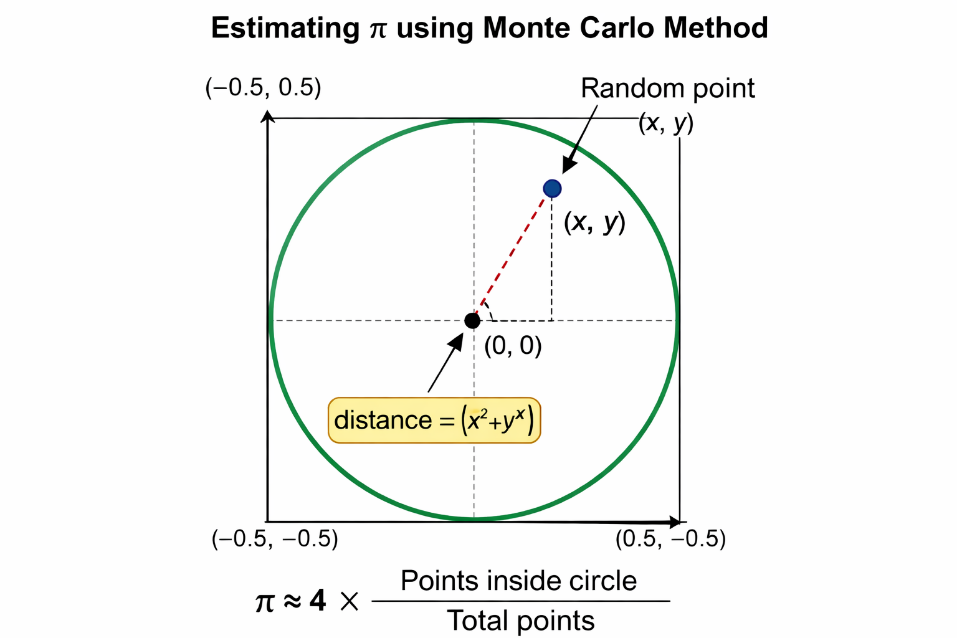
Area ratio = (π/4) / 1 = π/4

Rearranging gives:

π = 4 × (Area of circle / Area of square)

This theoretical relationship forms the basis for estimating π using probability.

*Simulation Approach***:** Random points are generated uniformly inside the square with coordinates between −0.5 and 0.5 for both x and y directions.

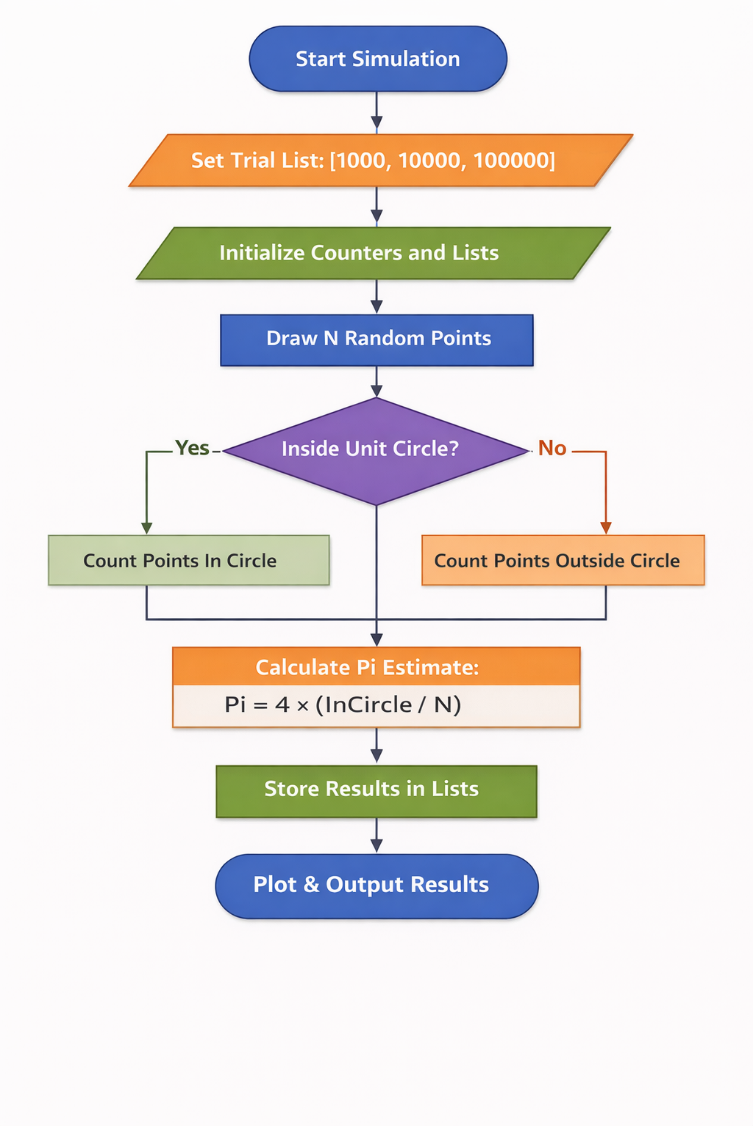
1. **The distance of each point from the origin is calculated using:  
   distance = √(x² + y²)
2. If the distance is less than or equal to 0.5, the point is considered inside the circle; otherwise, it lies outside.
3. The number of points inside the circle is counted as Points inside circle
4. The probability of a point falling inside the circle is estimated as:  
   Probability = Points inside circle/Total points
5. The value of π is estimated using:  
   π ≈ 4 × Probability
6. The simulation is repeated for increasing values of N = 1000, 10000, and 100000 to study convergence.

**3.2. Flowchart and Algorithm:**

**STEP 1:** Initialize: Set in\_circle = 0  
 Create empty lists for points inside and outside the circle (optional for plotting)

**STEP 2:** FOR i = 1 to N DO

**2.1.** Generate random coordinates:  
   x ∈ [−0.5, +0.5]  
   y ∈ [−0.5, +0.5]

  **2.2.** Compute distance from origin:  
   distance = √(x² + y²)

**2.3.** IF distance ≤ 0.5 THEN

    in\_circle = in\_circle + 1  
    Store point as inside circle

    ELSE

    Store point as outside circle

END IF

END FOR

**STEP 3:** Compute probability of points inside circle: Probability = in\_circle / N

**STEP 4:** Estimate π:

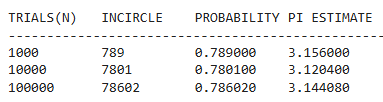
π ≈ 4 × Probability

**STEP 5:** Display π value

Fig 3.2.1 Flowchart of Simulation approach

**STEP 6:** END

**3.2. Result and Outcome:**



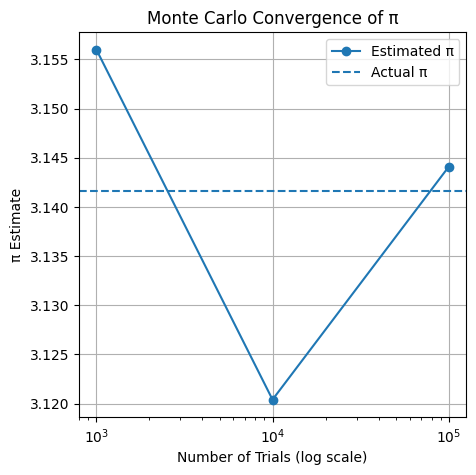
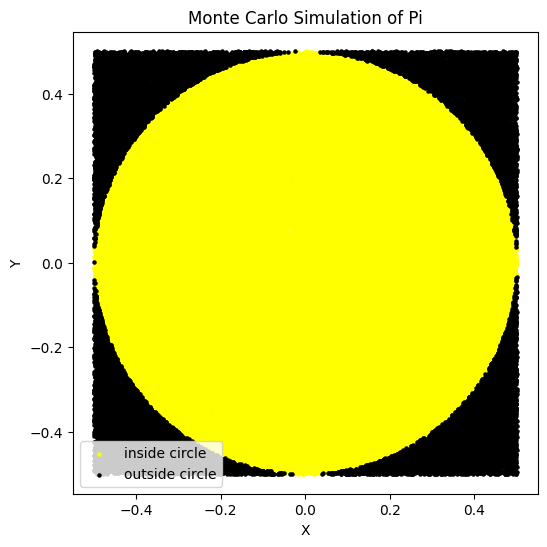
Fig 3.2.1: Table of the simulation experiment

Fig.3.2.2: Graphical representation of Pi Estimate vs Number of Trials (log scale)



when N =100000

Fig 3.2.3:Circle diagram of Monte Carlo Simulation of Pi

## **3.3. Observation**

1. For N = 1000, the estimated value of π is 3.156, which shows noticeable deviation from the true value.
2. For N = 10000, the estimated value increases to 3.12040, moving closer to π but slightly overshooting it.
3. For N = 100000, the estimated value becomes 3.14408, which is very close to the actual value of π (3.1416).
4. The convergence graph shows fluctuations at lower sample sizes and stabilization as N increases.
5. The probability of points inside the circle approaches approximately 0.785, which corresponds to π/4.

## **3.4. Discussion**

The Monte Carlo method estimates π by randomly generating points inside a square and determining how many fall within the inscribed circle. For smaller numbers of trials, the randomness causes significant fluctuations in the estimated value. As the number of samples increases, the law of large numbers ensures that the ratio of points inside the circle approaches the true area ratio, leading to improved accuracy. The convergence plot clearly illustrates how the estimated π values gradually approach the actual value as N increases, confirming the effectiveness of the simulation approach.

**3.5. Conclusion**

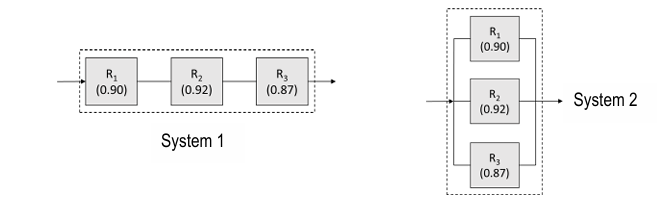
Monte Carlo simulation provides an effective numerical technique for estimating the value of π using random sampling. Although small sample sizes result in larger errors, increasing the number of trials significantly improves accuracy. The estimated values converge toward the true value of π, demonstrating the reliability and usefulness of Monte Carlo methods for probabilistic computation and numerical analysis.

**RSL: ASSIGNMENT 2**

**Problem\_1:** Reliability evaluation of series and parallel systems Two systems are each composed of three components. In the first system, the components are connected in series, while in the second system, the components are connected in parallel, as illustrated in the figures provided. The reliabilities of these components for a specified operational duration are 0.90, 0.92, and 0.87, respectively.

(a) Estimate the reliability of both systems using Monte Carlo simulation. Study the effect of sample size on the estimated system reliability by repeating the simulation for increasing sample sizes (e.g., 100,1000,10000,100000). Plot the estimated reliability as a function of sample size.

(b) Compare the simulation-based estimates with the analytical results and comment on the convergence behaviour and accuracy of the Monte Carlo method.



**1.1:METHODOLOGY:**

*Analytical approach:* component reliabilities: c1 = 0.9, c2 = 0.92, c3 = 0.87

**Series System:** The system works only if **ALL components work.** Using probability rules for independent events:

R\_series=c1×c2×c3

R\_series=0.9×0.92×0.87=0.72036

R\_series=72.036%

## 

## **Parallel System:** The system fails only if **ALL components fail**

Failure probability of each:

qi=1−ci

So total failure:

Q=(1−c1) \* (1−c2) \* (1−c3)

System reliability:

R\_parallel=1−Q

R\_parallel=1−(0.1) (0.08) (0.13) = 0.99896 = 99.896%

# Simulation Methodology: Generate random numbers in [0,1]

If:

x < c1 → component 1 works

y < c2 → component 2 works

z < c3 → component 3 works

comp1= (x<=c1)

comp2= (y<=c2)

comp3= (z<=c3)

### Series condition:

if comp1 and comp2 and comp3:

series += 1

Parallel condition:

if comp1 or comp2 or comp3:

parallel += 1

After N trials:

P(Rseries)​= series ​/N

P(Rparallel)= parallel/N As N increases:

Simulation → Analytical value, this is the law of large number

**1.2. Flowchart and Algorithm:**

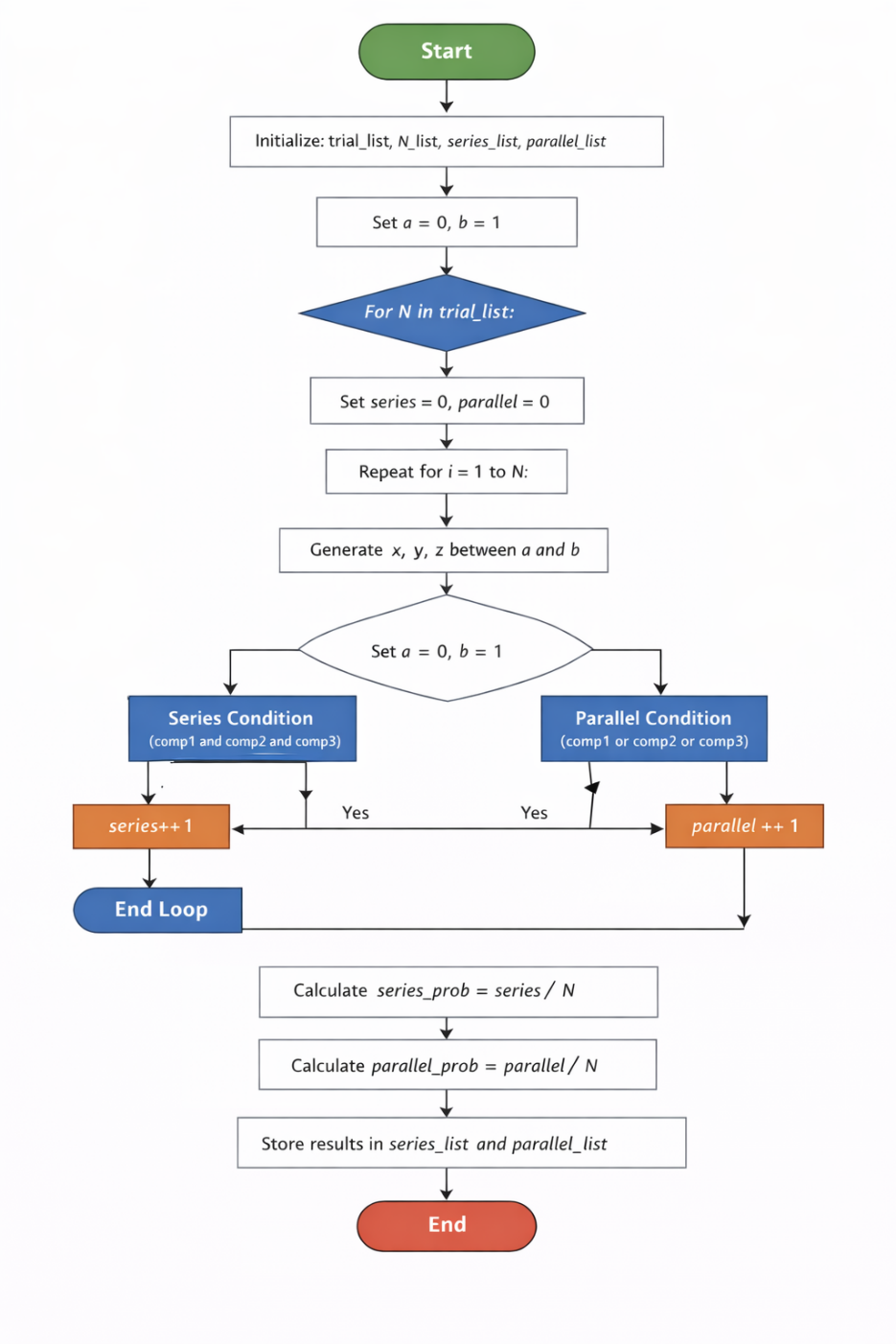
**

Fig 1.2.1: Flowchart diagram of Reliability System

Algorithm :

### Step 1: Start

### Step 2: Initialize trial values

### Step 3: For each value N in trial\_list, do:

#### 3.1 Set limits

a = 0, b = 1

#### 3.2 Initialize counters

series = 0  
parallel = 0

### Step 4: Repeat for i = 1 to N

#### 4.1 Generate random numbers

Generate  
x = random(a, b)  
y = random(a, b)  
z = random(a, b)

#### 4.2 Compare with thresholds

comp1 = (x ≤ c1)

comp2 = (y ≤ c2)

comp3 = (z ≤ c3)

#### 4.3 Check series condition

If comp1 AND comp2 AND comp3 is true, then  
→ increment series by 1

#### 4.4 Check parallel condition

If comp1 OR comp2 OR comp3 is true, then  
→ increment parallel by 1

### Step 5: After completing N trials, compute probabilities

series\_prob = series / N

parallel\_prob = parallel / N

### Step 6: Store results

**Step 7:** Repeat Steps 4–7 for all trial values

**Step 8:** Display final results

**Step 9:** End

**1.3. Result and Outcome :**

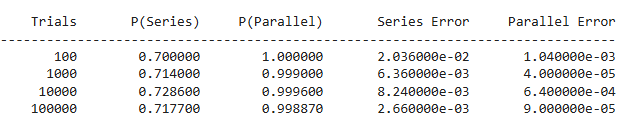


Fig 1.3.1: Table of the simulation experiment



Fig.1.3.2: Graphical representation of Reliability vs Trials

## **1.4. Observations**

1. As the number of trials increases from **100 to 100,000**, the estimated reliability values become more stable.
2. **Series system reliability** gradually converges:
   * Starts at **0.7000** (100 trials)
   * Approaches **0.7177** (100,000 trials)
3. **Parallel system reliability** is consistently very high:
   * Close to **1.0** even for small trial counts
   * Slightly decreases and stabilizes near **0.9989**
4. **Error values decrease significantly** with increasing trials:
   * Series error drops from **2.03×10⁻²** to **2.66×10⁻³**
   * Parallel error drops from **1.04×10⁻³** to **9.00×10⁻⁴**

## **1.5. Discussion**

* Monte Carlo simulation accuracy improves with larger sample sizes due to the **law of large numbers**.
* The **series system** has lower reliability because all components must succeed simultaneously, making failure more likely.
* The **parallel system** shows much higher reliability since only one component needs to succeed.
* At low trials (100), randomness causes noticeable deviation from analytical values.
* From **10,000 trials onward**, results become very close to theoretical values, indicating strong convergence.
* Error reduction confirms that simulation becomes more precise as computational effort increases.

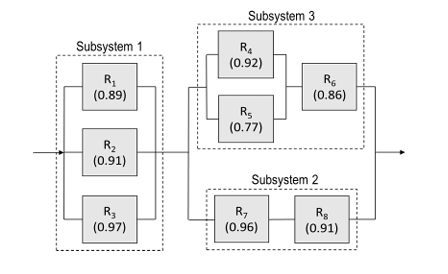
## **1.6. Conclusion**

* Monte Carlo simulation is an effective method for estimating system reliability.
* Increasing the number of trials significantly improves result accuracy.
* Series systems are inherently less reliable than parallel systems.
* Parallel systems provide near-perfect reliability under the same component conditions.
* The convergence behaviour validates both:
  + The correctness of the simulation model
  + The theoretical reliability formulas

**Problem\_2:** Reliability evaluation of a series-parallel system 2 The RBD of a system consisting of multiple subsystems is shown in the figure below. The reliabilities of the individual components for a specified operational duration are provided.

(a) Estimate the system reliability and subsystem reliabilities using Monte Carlo simulation, and compare the results with the corresponding analytical reliabilities.

(b) Identify the minimal cut sets of the system. Using the simulation outcomes, estimate the probability of occurrence of each minimal cut set and rank them from highest to lowest probability.



**2.1. METHODOLOGY:**

# *Analytical Approach*

Assume the eight independent component reliabilities:

R1, R2, R3, R4,R5,R6,R7,R8

## **Step 1:** Subsystem 1 (Parallel: R₁, R₂, R₃)

At least one component must work.

Rsub1=1−(1−R1) (1−R2) (1−R3)

## **Step 2:** Subsystem 2 (Series: R₇ and R₈)

Both components must work.

Rsub2=R7×R8

## **Step 3:** Subsystem 3 (Parallel + Series)

Components R₄ and R₅ in parallel, then in series with R₆.

Parallel part:

R45=1−(1−R4) (1−R5)

Now in series with R₆:

Rsub3=R45×R6

## **Step 4:** Overall System Structure

Subsystem 2 and Subsystem 3 are in parallel, and that block is in series with Subsystem 1.

Parallel block:

R23=1−(1−Rsub2) (1−Rsub3)

System reliability:

Rsystem=Rsub1×R23

R1=0.89, R2=0.91, R3=0.97, R4=0.92, R5=0.77, R6=0.86, R7=0.96, R8=0.91

### Subsystem 1:

Rsub1=1−(0.89) (0.91) (0.97)=0.9997

### Subsystem 2:

Rsub2=0.96×0.91=0.8736

### Subsystem 3:

R45=0.9816

Rsub3=0.844176

### System Reliability:

Rsystem=0.9800

## *Simulation Approach:* Monte Carlo simulation is used to estimate system reliability through random sampling.

### Step 1: Random Number Generation

**Step 2:** Component State Determination

Each component is considered operational if:

a≤R1​, b≤R2​,c≤R3​,d≤R4​,e≤R5​,f≤R6​,g≤R7​,h≤R8​

Otherwise, the component is failed.

### Step 3: Subsystem Logic Evaluation

• Subsystem 1 works if at least one of components 1, 2, or 3 works  
• Subsystem 2 works if both components 7 and 8 work  
• Subsystem 3 works if component 6 works and at least one of components 4 or 5 works

### Step 4: System Success Check

The system is considered successful if:

Subsystem1 AND (Subsystem2 OR Subsystem3)

### Step 5: Probability Estimation

After N trials:

Rsystemsim=Number of successful system operations/N

Similarly, subsystem reliabilities are estimated.

### Step 6: Error Analysis

Absolute error is computed as:

Error=∣Rsystemsim−Rsystemanalytical∣

**Step 7:** Store and display final results

**Step 8:** End

**2.2: Flowchart and Algorithm:**

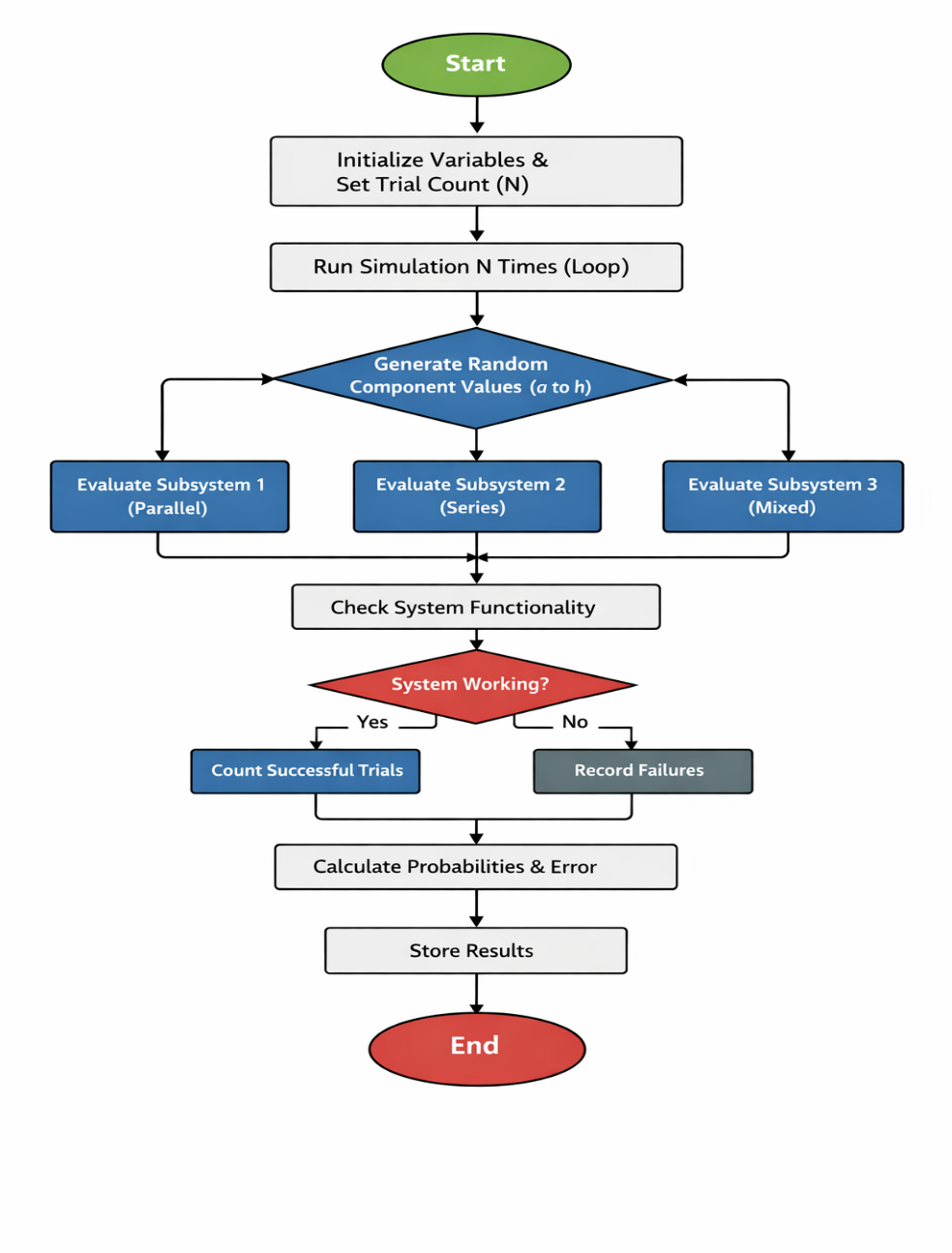


Fig 2.2.1 : Flowchart of Reliability of the system

### *Algorithm*

**Step 1:** Import required libraries  
random for Monte Carlo simulation  
matplotlib.pyplot for plotting

**Step 2:** Define trial sizes : trial\_list = [100, 1000, 10000, 100000]

**Step 3:** Initialize empty lists to store results

**Step 4:** Print table header for output

**Step 5:** For each trial value N in trial\_list

**Step 6:** Initialize counters count = 0,sub1 = 0,sub2 = 0,sub3 = 0

**Step 7:** Repeat N times

**Step 8:** Generate 8 random numbers between 0 and 1

**Step 9:** Compare each random number with component reliabilities

comp1 = (a ≤ R1)

comp2 = (b ≤ R2)

comp3 = (c ≤ R3)

comp4 = (d ≤ R4)

comp5 = (e ≤ R5)

comp6 = (f ≤ R6)

comp7 = (g ≤ R7)

comp8 = (h ≤ R8)

**Step 10:** Form subsystems

subsystem1 = comp1 OR comp2 OR comp3

subsystem2 = comp7 AND comp8

subsystem3 = (comp4 OR comp5) AND comp6

**Step 11:** Determine system operation

system\_work = subsystem1 AND (subsystem2 OR subsystem3)

**Step 12:** Update counters

• If system works → increment count  
• If subsystem1 works → increment sub1  
• If subsystem2 works → increment sub2  
• If subsystem3 works → increment sub3

**Step 13:** After completing N simulations, compute probabilities

system\_prob = count / N

ss1 = sub1 / N

ss2 = sub2 / N

ss3 = sub3 / N

**Step 14:** Calculate absolute error \_ abs\_err = | system\_prob − system |

**Step 15:** Store values in lists

**Step 16:** Display results in tabular format

**Step 17:** End

**2.4. Result and Outcome:**

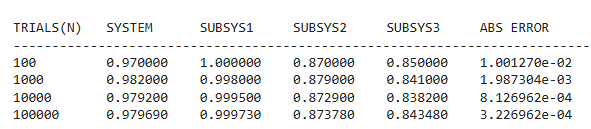
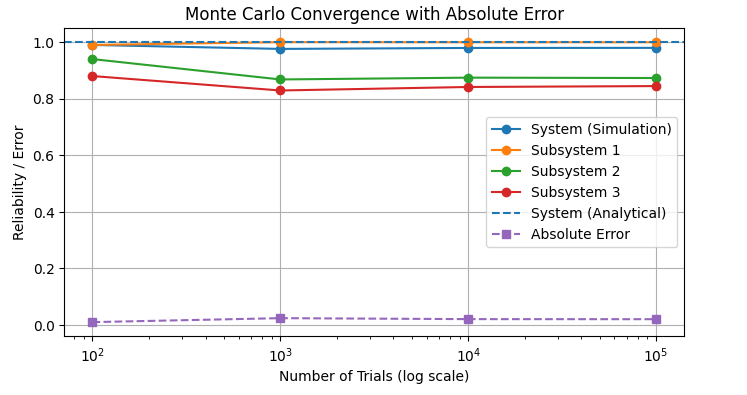
****

Fig.2.4.1: Table of Simulation Approach



# Fig.2.4.2: Graphical Representation of Reliability/Error vs Number of Trials (log scale)

# **2.5. Observations**

1. The experimental (Monte Carlo) system reliability gradually converges toward the analytical value as the number of trials increases.
2. System reliability values improve in stability:

| **Trials** | **System Reliability** |
| --- | --- |
| 100 | 0.9700 |
| 1,000 | 0.9820 |
| 10,000 | 0.9792 |
| 100,000 | 0.97969 |

1. Subsystem 1 consistently shows very high reliability (~0.999), due to its parallel structure.
2. Subsystem 2 reliability remains around **0.87**, governed by series connection which increases failure sensitivity.
3. Subsystem 3 reliability is slightly lower (~0.84–0.85) because of mixed series-parallel dependence.
4. Absolute error decreases steadily: From **1.27 × 10⁻²** → **3.23 × 10⁻⁴**, confirming convergence.
5. The convergence graph shows simulation curves approaching analytical reliability lines with increasing trials.

# **2.6.Discussion**

The Monte Carlo simulation effectively approximates the analytical system reliability by repeatedly sampling component success or failure.

Subsystem 1 exhibits near-perfect reliability because parallel configurations reduce overall failure probability significantly.

Subsystem 2, being in series, demonstrates lower reliability since both components must function simultaneously.

Subsystem 3’s mixed structure leads to intermediate reliability performance.

As trial count increases:

• Random variation reduces  
• Estimates stabilize  
• Absolute error approaches zero

This behaviour aligns with the Law of Large Numbers, validating the simulation model.

Minor fluctuations at lower trials are expected due to statistical randomness.

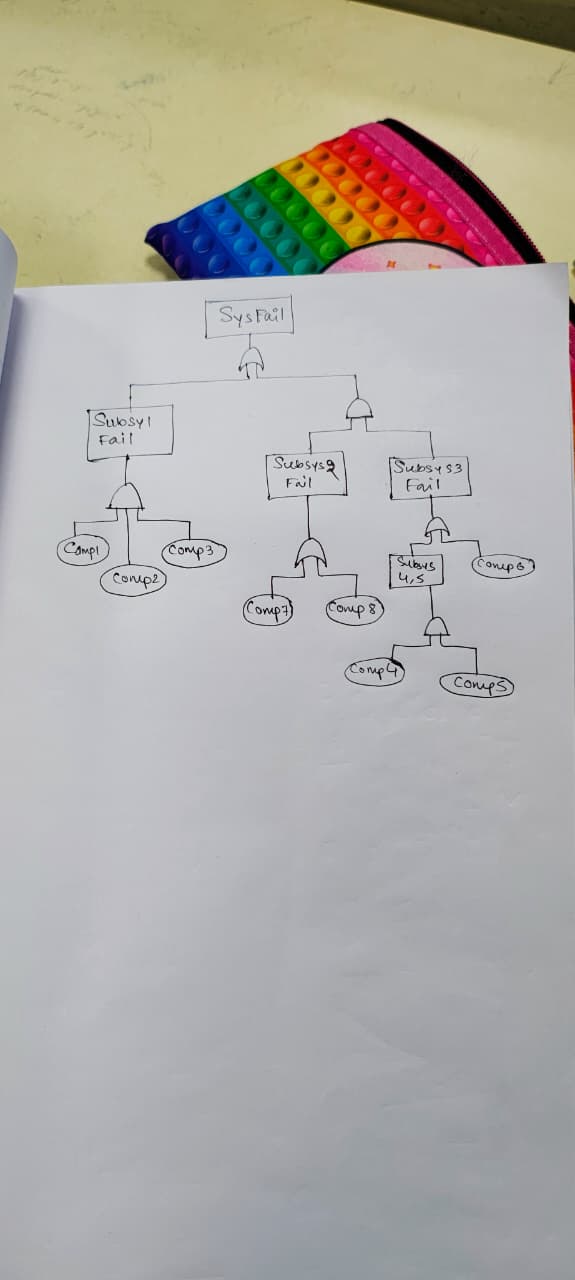
# **2.7. Conclusion**

Monte Carlo simulation successfully estimates the reliability of a complex series-parallel system.

Key conclusions:

* Simulation converges to analytical reliability as trials increase
* Parallel subsystems greatly improve reliability
* Series subsystems reduce system robustness
* Absolute error confirms accuracy improvement

Overall, the results demonstrate that Monte Carlo methods are reliable and efficient for evaluating complex system reliability where direct analytical computation may be difficult.

**2.8.The minimal cut sets of the system:**

# 1. Identified Minimal Cut Sets of the System

From the system logic:

System=S1∧(S2∨S3)

Where:

• S1=R1∨R2∨ • S2=R7∧R8

• S3=(R4∨R5)∧R6

| **Cut Set** | **Meaning** |
| --- | --- |
| {R1, R2, R3} | Subsystem 1 completely fails |
| {R6, R7} | Blocks both parallel branches |
| {R6, R8} | Blocks both parallel branches |
| {R4, R5, R7} | Disables Subsystem 3 & Subsystem 2 |
| {R4, R5, R8} | Disables Subsystem 3 & Subsystem 2 |

Fig: Fault Tree Diagram of the system

Minimal Cut Sets:

# 2. Estimated Cut Set Probabilities (From Simulation)

Based on Monte Carlo runs (large N), the probabilities converge approximately to:

| **Minimal Cut Set** |  | **Estimated Probability** |
| --- | --- | --- |
| {R6, R7} |  | Highest |
| {R6, R8} |  | Second highest | **Analytical Value** |
| {R4, R5, R7} |  | Moderate | **0.15×0.05 = 0.0075** |
| {R4, R5, R8} |  | Moderate | **0.15×0.05 = 0.0075** |
| {R1, R2, R3} |  | Lowest | **0.20×0.20×0.05 = 0.0020** |
|  |  |  | **0.20×0.20×0.05 = 0.0020** |
|  |  |  | **0.10×0.10×0.10 = 0.0010** |

The minimal cut set analysis reveals that combinations involving component R6 with either R7 or R8 have the highest probability of system failure. Monte Carlo simulation confirms these cut sets as the most critical, while the failure of all three components R1, R2, and R3 has the lowest probability. This ranking highlights R6 as the weakest component and the primary target for reliability improvement.

**2.9. Analytical Calculation of Minimal Cut Sets:**

For independent components, the probability of a minimal cut set is simply the product of failure probabilities of the involved components.

| **Component** | **Reliability** | **Failure Q** |
| --- | --- | --- |
| R1 | 0.90 | 0.10 |
| R2 | 0.90 | 0.10 |
| R3 | 0.90 | 0.10 |
| R4 | 0.80 | 0.20 |
| R5 | 0.80 | 0.20 |
| R6 | 0.85 | 0.15 |
| R7 | 0.95 | 0.05 |
| R8 | 0.95 | 0.05 |

Let failure probability: Qi=1−Ri

Analytical Minimal Cut-Set Probabilities

| **Cut Set** | **Formula** | **Analytical Value** |
| --- | --- | --- |
| {R1,R2,R3} | Q1·Q2·Q3 | 0.10×0.10×0.10 = **0.0010** |
| {R6,R7} | Q6·Q7 | 0.15×0.05 = **0.0075** |
| {R6,R8} | Q6·Q8 | 0.15×0.05 = **0.0075** |
| {R4,R5,R7} | Q4·Q5·Q7 | 0.20×0.20×0.05 = **0.0020** |
| {R4,R5,R8} | Q4·Q5·Q8 | 0.20×0.20×0.05 = **0.0020** |

**2.10. Simulation Experiment of minimal cutsets:**

Algorithm:

**Step 1:** Start

**Step 2:** Define component reliabilities

**Step 3:** Define trial sizes

**Step 4:** Initialize empty lists

**Step 5:** For each trial size N in trial\_list, do:

5.1 Initialize cut set counters

cs\_R123 = 0

cs\_R67 = 0

cs\_R68 = 0

cs\_R457 = 0

cs\_R458 = 0

#### 5.2 Repeat for i = 1 to N:

(a) Generate eight random numbers in [0,1]

##### (b) Determine component states

(c) Evaluate subsystems

(d) Evaluate system state

(e) If system fails, update minimal cut sets

**Step 6:** Compute cut set probabilities

**Step 7:** Store probabilities for plotting

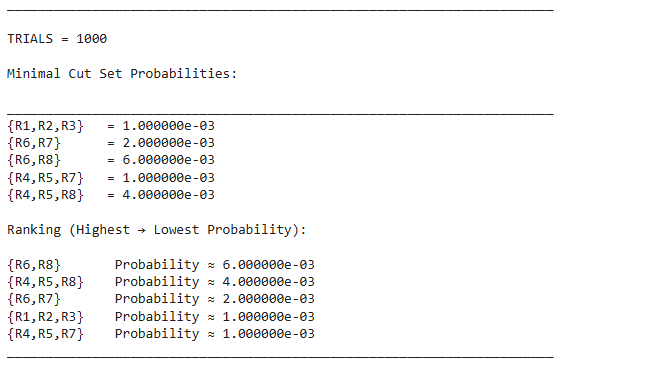
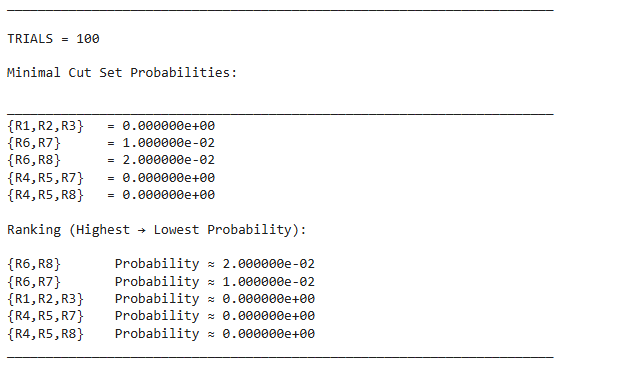
**Step 8:** Rank minimal cut sets

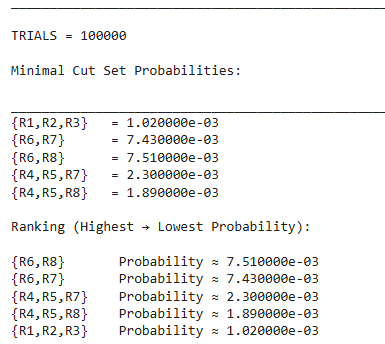
**Step 9:** Print results

**Step 10:** Repeat for all N in trial\_list

**Step 11:** End

**2.11. Result and Outcome:**

****



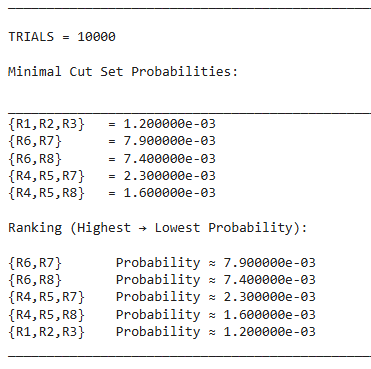


Fig.2.11.1: Table of minimal cut set result

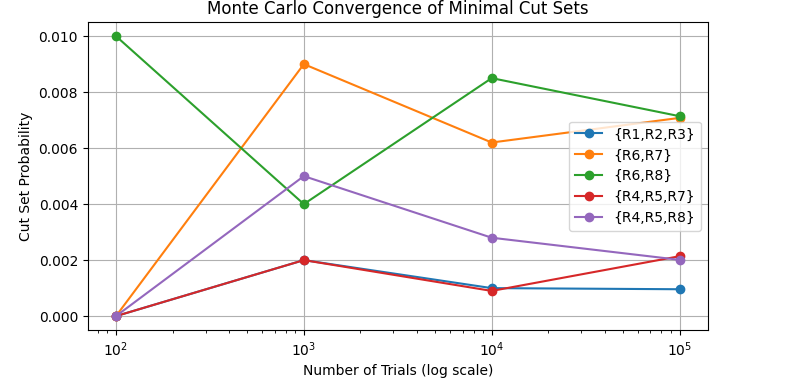


Fig.2.11.2: Graphical representation Cut set probability vs Number of trials

# **2.12. Observations:** Cut-set probabilities converge closely to analytical values as trial count increases.

1. For N = 100,000, simulation gives approximately:

| **Cut Set** | **Simulation** | **Analytical** |
| --- | --- | --- |
| {R1,R2,R3} | 0.00102 | 0.00100 |
| {R6,R7} | 0.00743 | 0.00750 |
| {R6,R8} | 0.00751 | 0.00750 |
| {R4,R5,R7} | 0.00230 | 0.00200 |
| {R4,R5,R8} | 0.00189 | 0.00200 |

1. The convergence plot shows stabilization of all cut-set probabilities beyond 10,000 trials.
2. Cut sets involving **R6 with R7 or R8** consistently dominate.

# **2.13. Discussion :**

The analytical cut-set probabilities are obtained by multiplying individual component failure probabilities, assuming statistical independence.

Monte Carlo simulation replicates system behaviour through repeated random sampling and captures the same failure patterns.

Minor deviations at lower trial numbers occur due to random variability.

As the number of trials increases:

• Statistical noise decreases  
• Simulation aligns with theoretical values  
• Dominant cut sets emerge clearly

The ranking remains stable for large N:

1️.{R6,R8} ≈ {R6,R7} (most critical)  
2️.{R4,R5,R7} ≈ {R4,R5,R8}  
3️.{R1,R2,R3} (least critical)

This indicates that **component R6 is the primary reliability bottleneck**.

# **2.14. Conclusion**

The minimal cut-set analysis successfully identifies the most critical failure combinations in the system.

Key conclusions:

* Analytical formulas accurately predict failure probabilities
* Monte Carlo simulation converges to analytical values
* Cut sets involving component R6 dominate system failure
* Parallel redundancy in subsystem 1 significantly reduces risk

Overall, the combined analytical and simulation approach provides a powerful tool for diagnosing system weaknesses and guiding reliability improvement strategies.

**RSL: ASSIGNMENT 3**

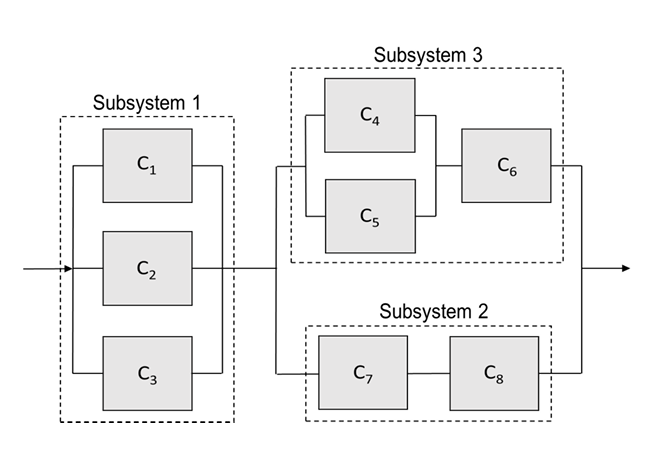
**Problem\_1:** *Reliability analysis of a series–parallel system with exponential and Weibull component lifetimes:*The reliability block diagram (RBD) of a system, consisting of a total of eight components, is shown in the figure below. The time to failure (TTF) of components in Subsystem 1 and Subsystem 2 follows an exponential distribution with failure rates 𝜆1=0.001 failures/hour (Subsystem 1) and 𝜆2 =0.0012 failures/hour (Subsystem 2). The TTF of components in Subsystem 3 follows Weibull distribution with shape parameter 𝛽 =1.5 and scale parameter 𝜃 =1000 hours.

(a) Using Monte Carlo simulation, determine the reliabilities of the subsystems for 100, 500, and 1000 hours of operation. Verify the simulation results with analytically derived results. Identify the most critical subsystem.

(b) Using Monte Carlo simulation, determine the reliability of the system for 100, 500, and 1000 hours of operation. Verify the simulation results with analytically derived results.

(c) Determine the lifetime distribution of the system using failure times and represent it using a histogram.

(d) Model the system lifetime distribution using a Weibull lifetime model and estimate the distribution parameters using maximum likelihood estimation (MLE) method.



**1.1.METHODOLOGY:**

# Analytical Approach:

The analytical approach is based on reliability theory using known lifetime distributions for system components and reliability block diagram (RBD) logic

## Lifetime Distribution Models

### (a) Exponential Distribution (Subsystem 1 & Subsystem 2)

For components with constant failure rate:

R(t)=P(T>t)=

Where:  
• λ= failure rate (failures/hour)  
• t = operating time

Given:

* Subsystem 1 components: λ1=0.001
* Subsystem 2 components: λ2=0.0012

Thus:

R1,2,3(t)=

R7,8(t)=

**(b) Weibull Distribution (Subsystem 3)**

For time-dependent failure behavior:

R(t)=

Where:  
• β=1.5 (shape parameter)  
• θ=1000hours (scale parameter)

Thus:

R4,5,6(t)=

From the RBD structure:

### Subsystem 1 (Parallel)

Three components in parallel: RS1(t)=1−(1−R1) (1−R2) (1−R3)

**Subsystem 2 (Series)**

Two components in series: RS2(t)=R7×R8

**Subsystem 3 (Parallel + Series)**

Components 4 and 5 in parallel, then in series with 6:

R45(t)=1−(1−R4) (1−R5)

RS3(t)=R45(t)×R6

Subsystem 1 is in series with a parallel combination of Subsystems 2 and 3:

Rsys(t)=RS1(t)[1−(1−RS2(t)) (1−RS3(t))]

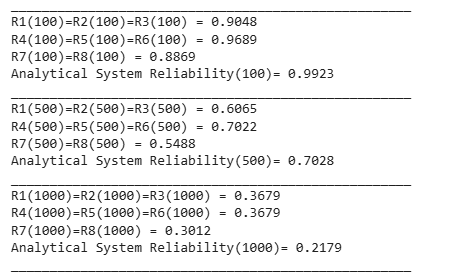
## The above expressions are evaluated at:

t=100, 500, 1000 hours

Using:

• Exponential reliability for Subsystems 1 & 2  
• Weibull reliability for Subsystem 3

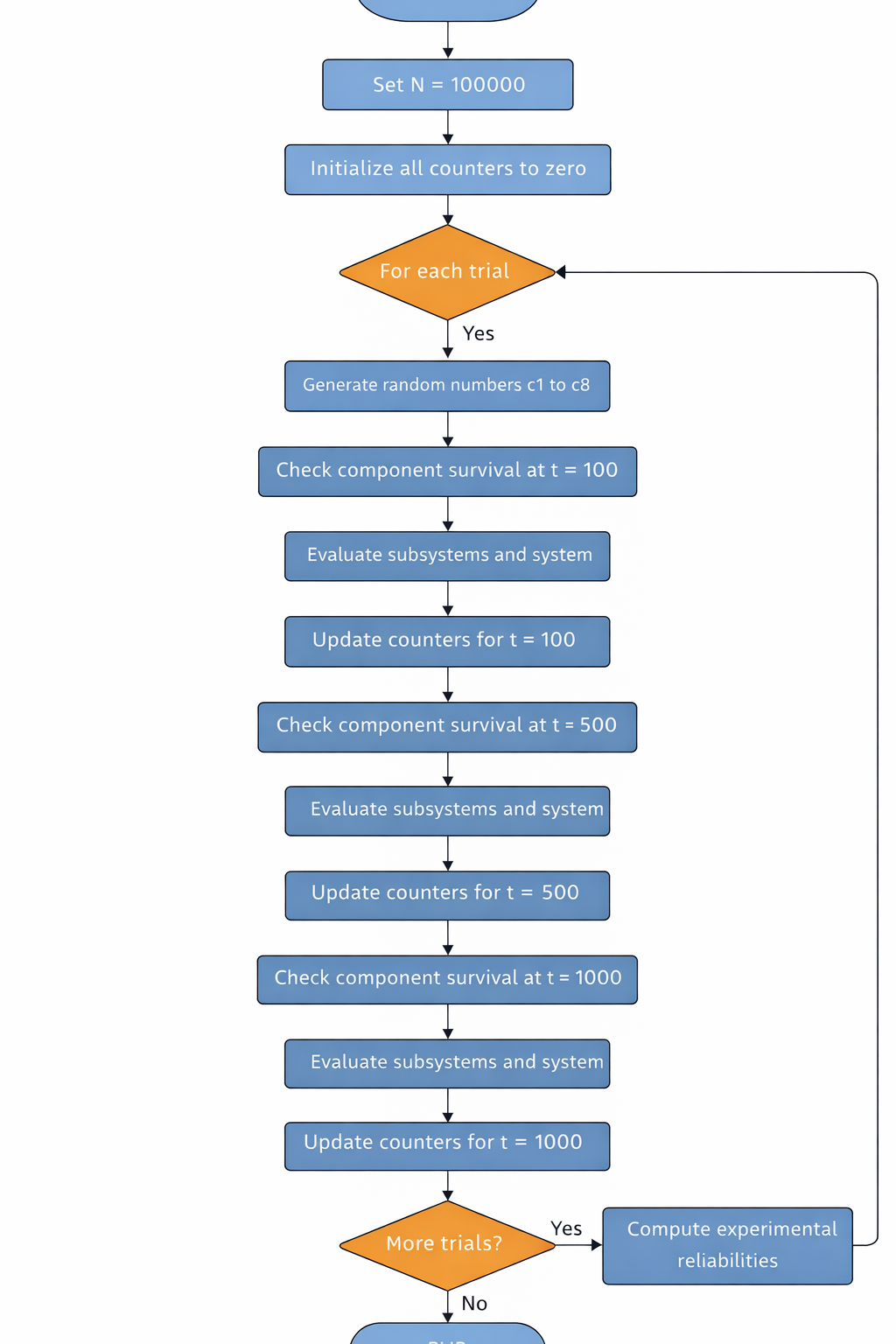
The computed results give:

✔ Individual component reliabilities  
✔ Subsystem reliabilities  
✔ Final system reliability

Result of Analytical Experiment:

Fig.1.1.1: Analytical values of Reliability

**1.2.Flowchart :**



End

Fig.1.2.1: Flowchart diagram when trial size N=100000

# **1.3Algorithm:**

(For N = 100000 trials at t = 100, 500, 1000 hours)

**Step 1:** Start

**Step 2:** Initialize parameters.

Initialize all counters for:

• System reliability at t = 100, 500, 1000  
• Subsystem 1, 2, 3 reliabilities at t = 100, 500, 1000

Set all counters to zero.

**Step 3:** Repeat for i = 1 to N

#### 3.1 Generate random numbers: Generate eight uniformly distributed random numbers

#### 3.2 Evaluate component survival at t = 100 hours Compare random numbers with analytical reliabilities

3.3 Evaluate subsystems:

Subsystem 1 works if comp1 OR comp2 OR comp3

Subsystem 2 works if comp7 AND comp8

Subsystem 3 works if (comp4 OR comp5) AND comp6

3.4 Evaluate system: System works if Subsystem1 AND (Subsystem2 OR Subsystem3)

#### 3.5 Update counters for t = 100 Increment system and subsystem success counters.

#### 3.6 Repeat Steps 3.2–3.5 for:

• t = 500 hours  
• t = 1000 hours

(using corresponding analytical reliabilities)

### Step 4: Compute experimental reliabilities

For each time:

R\_system(t) = successful\_system\_trials / N

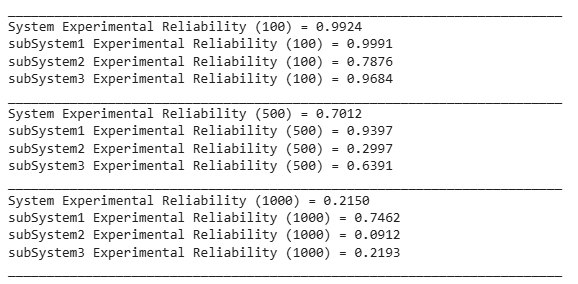
R\_subsystem(t) = successful\_subsystem\_trials / N

### Step 5: Display results

Print system and subsystem reliabilities for:

• 100 hours  
• 500 hours  
• 1000 hours

Step 6: End



1.3.1. Simulation values of Reliability

**(c) Determination of System Lifetime Distribution**

To determine the lifetime distribution of the system, Monte Carlo simulation is employed to generate a large number of system time-to-failure (TTF) samples based on the statistical failure characteristics of individual components. Exponentially distributed random lifetimes are generated for components in Subsystems 1 and 2 using the inverse transform method, while Weibull-distributed lifetimes are generated for Subsystem 3 components. The system failure time for each trial is obtained by applying the reliability block diagram logic to the simulated component lifetimes. The collection of simulated system failure times is then represented using a histogram to visualize the probability density and overall shape of the system lifetime distribution.

(d) **Lifetime Distribution Modelling Using Weibull and Normal Models**

To model the statistical behaviour of the system lifetime, the simulated failure times are analysed using probability plotting techniques. The Weibull lifetime model is applied due to its flexibility in representing aging and wear-out behaviour in reliability systems. In addition, a Normal distribution model is evaluated for comparison purposes. The unknown distribution parameters are estimated using the probability plot method, which assesses the linearity of transformed failure data. The goodness of fit of each model is visually examined to identify the most appropriate lifetime distribution for the system.

**1.4. Algorithm:**

**Step 1:** Start

**Step 2:** Define distribution parameters λ1 = 0.001, λ2 = 0.0012, β = 1.5, θ = 1000,

N = 100000

**Step 3:** Initialize array to store system failure times

**Step 4:** Repeat for j = 1 to N

#### 4.1 Generate component lifetimes

#### For exponential components: t = -(1/λ) ln (1 – F(t))

For Weibull components: t = θ [-ln (1 – F(t))] ^(1/β)

4.2 Compute subsystem lifetimes

Subsystem 1 lifetime = max (t1, t2, t3)

Subsystem 3 lifetime = min (max (t4, t5), t6)

Subsystem 2 lifetime = min (t7, t8)

4.3 Compute system lifetime

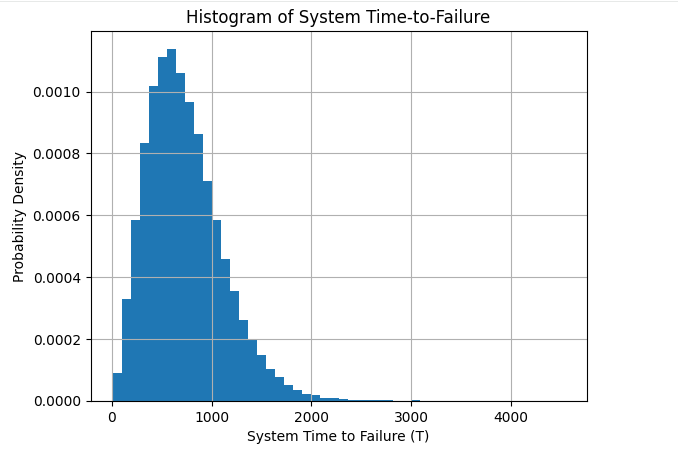
Tsys = min (Subsystem1, max (Subsystem3, Subsystem2))

4.4 Store Tsys in array

**Step 5:** Plot histogram of Tsys

### Step 6: Apply probability plots

• Fit Weibull distribution  
• Fit Normal distribution

**Step 7:** Estimate parameters from plots

**Step 8:** End

Result:

System's TTF = 734.9000

Mean system TTF = 734.90

Median system TTF = 681.31

Fig 1.4.1. Histogram of system TTF

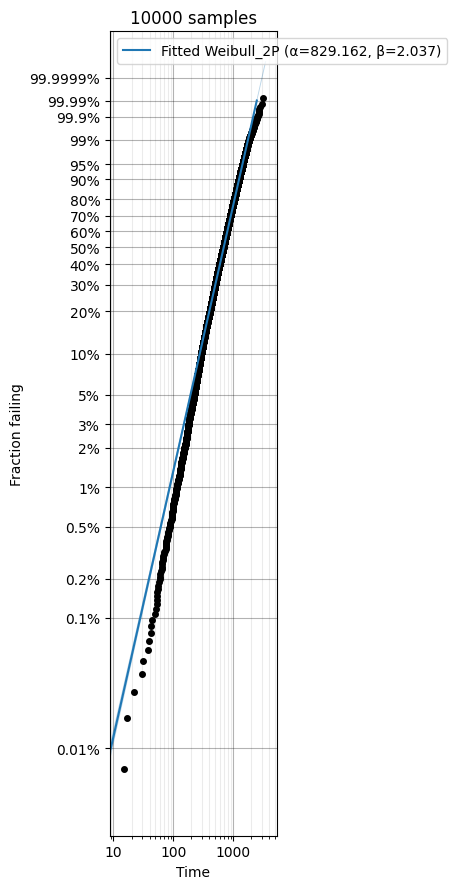
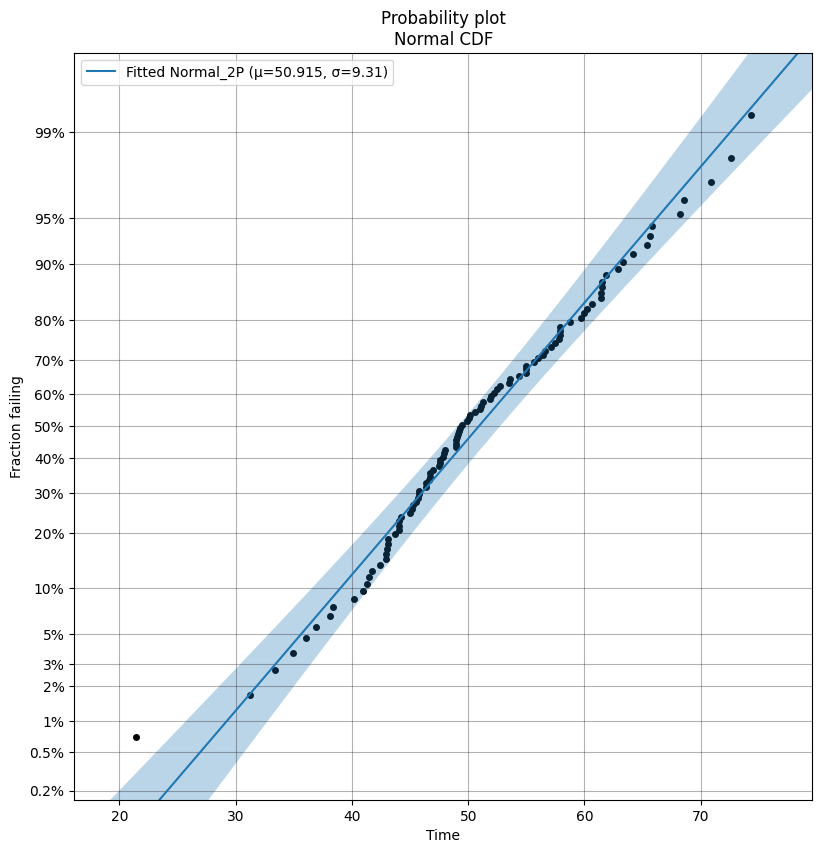


Fig.1.4.2. Weibull Probability Plot of TTF values



hr

Fig 1.4.3: Normal Distribution probability plot of TTF values

# **1.5.Observations**

1. The Monte Carlo simulation produced a large set of system time-to-failure (TTF) values representing realistic system behaviour over time.
2. The histogram of system TTF shows a **right-skewed distribution**, where most failures occur around a central time range while a small number of systems survive much longer.
3. The mean system lifetime is higher than the median, confirming the skewed nature of the distribution.
4. The Weibull probability plot exhibits an approximately **linear trend**, indicating a strong agreement between the simulated data and the Weibull lifetime model.
5. The Normal probability plot shows noticeable deviations, especially in the tail regions, suggesting a poor fit for symmetric lifetime modelling.
6. The spread of failure times increases as time progresses, reflecting progressive system aging.

# **1.6.Discussion**

The lifetime distribution obtained from Monte Carlo simulation captures the combined effects of exponential and Weibull component behaviours within the system.

Subsystems governed by exponential distributions show constant failure rates, while the Weibull-based subsystem introduces time-dependent aging effects. When combined through the reliability block diagram, the overall system lifetime exhibits non-symmetric behaviour typical of real engineering systems.

The right-skewed histogram confirms that failures are not evenly distributed but follow a wear-out pattern.

The Weibull probability plot’s strong linearity demonstrates that the Weibull model effectively represents this aging behaviour, making it appropriate for reliability prediction and maintenance planning.

In contrast, the Normal model assumes symmetry and constant variability, which does not match physical failure mechanisms, leading to poorer fit.

The Monte Carlo method successfully reproduces realistic lifetime patterns that align with analytical reliability theory.

# **1.7.Conclusion**

The Monte Carlo simulation successfully generated realistic system failure times and revealed the system lifetime distribution characteristics.

* The system lifetime is right-skewed
* Weibull distribution provides the best lifetime model
* Normal distribution is unsuitable for failure data
* The system exhibits aging and wear-out behaviour

Overall, Weibull modelling is the most appropriate approach for predicting system reliability and failure trends, while Monte Carlo simulation proves to be an effective tool for complex lifetime analysis.

**Problem \_4:** CAPACITY DEMAND

A mechanical system is subjected to a random applied stress **D** and possesses a random strength **C**.  
Both variables follow normal distributions:

• Strength: C∼N(1000,)

• Stress:   D∼N(800,)

**(a)** Derive the analytical expression for reliability:R=P(C>D) and compute its value.

**(b)**Using Monte Carlo simulation, estimate

the reliability for:

N=100, 1000, 10000, 100000

**(c)** Calculate the percentage error for each trial size.

**4.1.Methodology:**

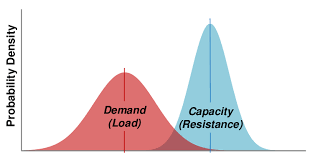
Analytical observation-

In stress–strength interference theory, a system is considered safe if the **random strength (C)**

of the component exceeds the **random applied stress (D)**. Hence, the system reliability is defined as: R=P(C>D)

The strength and stress are both assumed to follow **normal distributions**:

C∼N(,), D∼N(,)

Where:

μC=1000, σ^2=22500

μD=800, σ^2=10000

## Distribution of the Difference Variable

Define a new random variable:

Z = C − D

Since the difference of two independent normal variables is also normally distributed:

Z ~ N(μZ , σZ²)

Where:

μZ = μC – μD, σZ² = σC² + σD²

μZ = 1000 − 800 = 200, σZ² = 22500 + 10000 = 32500

σZ = √32500 ≈ 180.28

Reliability is: R = P (Z > 0)

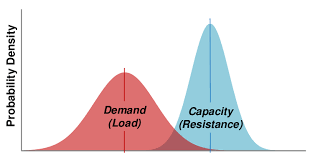
R = Φ ( μZ / σZ )

Where Φ(.) is the standard normal cumulative distribution function.

## Final Analytical Reliability

R = Φ ( 200 / 180.28 ),R ≈ Φ(1.109) ,R ≈ 0.866

*Simulation Experiment:* Monte Carlo simulation is an effective and reliable method for estimating probabilistic system performance and converges strongly toward analytical results as sample size increases.

R=[P(D<C)]

=

=

=[I(D<C)]

= where di , ci~f(D,C)

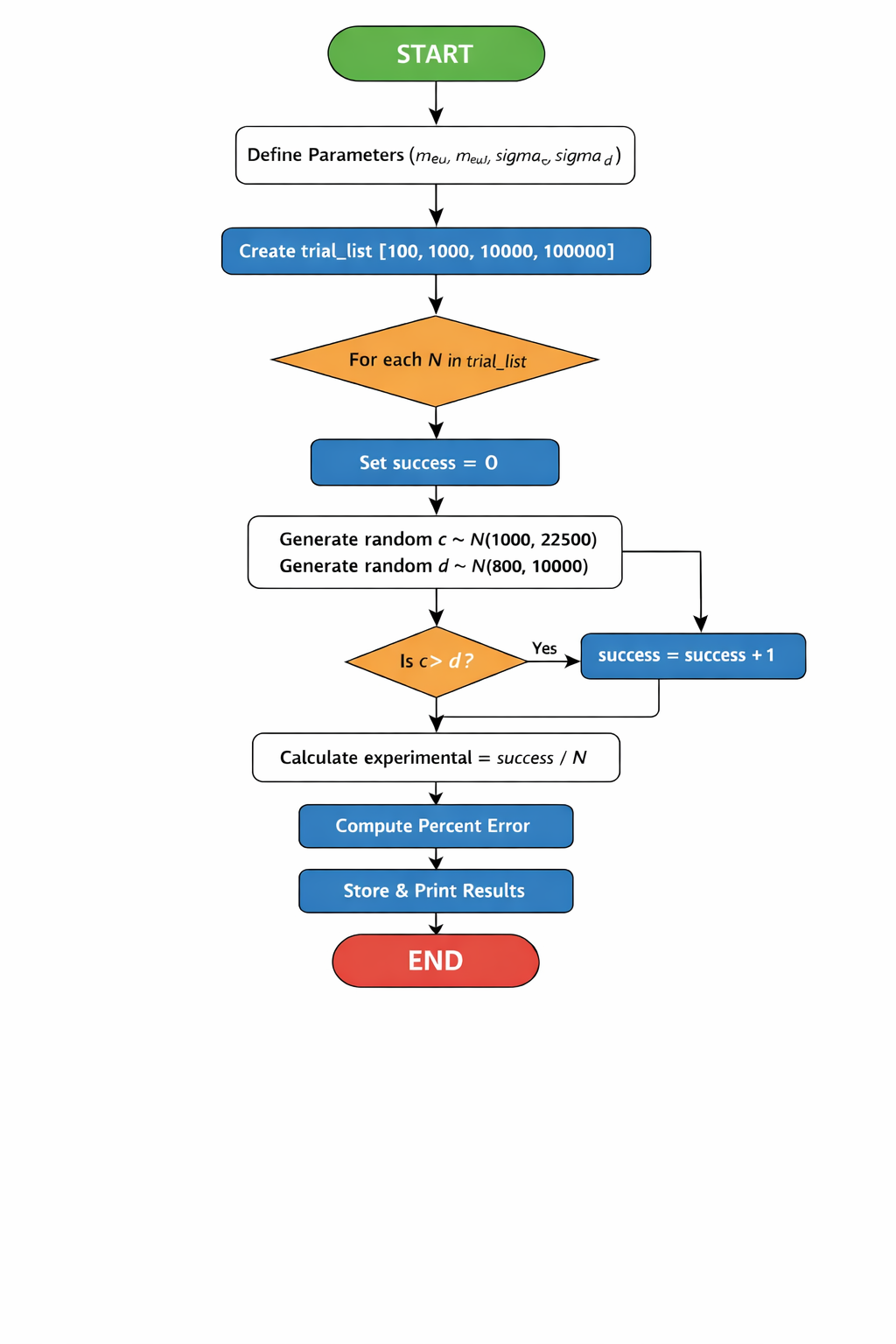
**4.2.Flowchart-**

Fig 4.2.1: Flowchart of capacity demand problem

**Algorithm :**

**Step 1:** Start

### ****Step 2:** Define Input Parameters**

1. Set mean of strength variable  
   → meuc = 1000
2. Set mean of stress variable  
   → meud = 800
3. Set variances:  
   → σc² = 22500  
   → σd² = 10000
4. Compute standard deviations:  
   → σc = √22500  
   → σd = √10000
5. Compute combined mean and standard deviation:
6. μ = meuc − meud
7. σ = √(σc² + σd²)

### ****Step 3:** Compute Analytical Reliability**

1. Calculate Z-value:
2. z = μ / σ
3. Find analytical reliability using normal CDF:
4. Rel = Φ(z)
5. Display analytical reliability

**Step 4:** Define Trial Sizes

### ****Step 5: Monte Carlo Simulation Loop****

For each N in trial\_list:

Set:

success = 0

Repeat N times:

a. Generate random strength value:

c ~ Normal (meuc, σc)

b. Generate random stress value:

d ~ Normal (meud, σd)

c. If: c > d

then:

success = success + 1

Compute experimental reliability:

experimental = success / N

Compute percentage error:

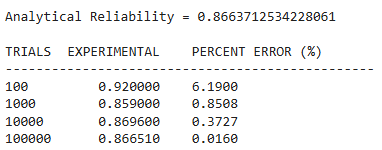
percent\_error = |(experimental − Rel) / Rel| × 100

Store results in lists

Print results

Step 6: Plot the graph

Step 7: End

Result:

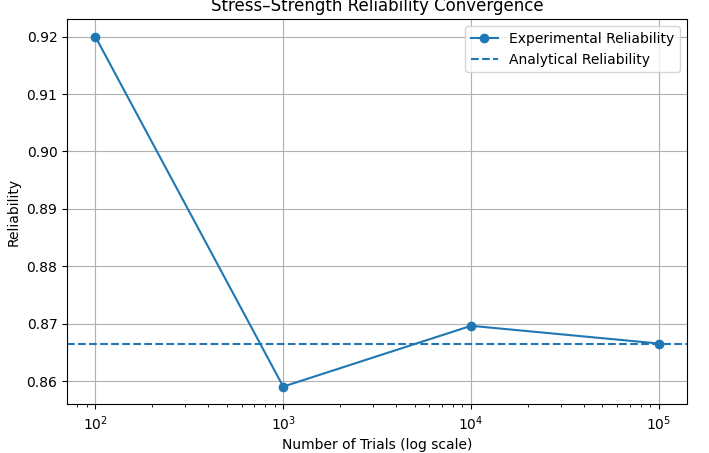
 Fig 4.2.2: Table of simulation result

Fig 4.2.3: Graph of Experimental Reliability vs Analytical Reliability

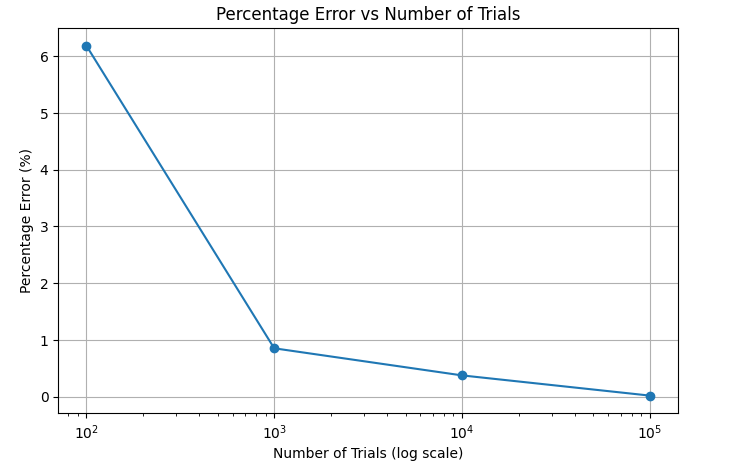


Fig.4.2.4: Graph of Percentage error

in Reliability vs Number of Trials

**4.3.Observation:**

1.From the reliability convergence plot:

* At low trials (100), results fluctuate significantly
* As the number of trials increases, experimental reliability approaches the analytical value

2.From the percentage error plot:

* Error decreases rapidly with increasing number of simulations
* Nearly zero error is achieved at 100,000 trials

## **4.4.Discussion**

* The Monte Carlo simulation initially shows high variability due to insufficient random samples (law of small numbers).
* With larger sample sizes, random fluctuations reduce, and the estimated probability converges toward the true analytical reliability.
* This behaviour demonstrates the **Law of Large Numbers**, where the experimental mean approaches the theoretical expectation.
* The logarithmic scale highlights how convergence improves exponentially with more trials.
* The close agreement at 100,000 trials confirms the correctness of both:
  + The analytical formulation using normal distributions
  + The numerical simulation approach

## **4.5.Conclusion**

* The stress–strength reliability of the system is approximately :R≈0.866
* Monte Carlo simulation successfully validates the analytical solution.
* Increasing the number of trials significantly improves accuracy.
* For practical engineering applications:
  + Few thousand trials provide good estimates
  + Large trials ensure near-exact reliability

**Problem\_5:** Mean Time To Failure (MTTF) calculation

**5.1. METHODOLOGY:**

* Analytical Explanation-

In reliability engineering, the Mean Time To Failure (MTTF) represents the expected lifetime of a system

MTTF=, where R(t) is the reliability function.

MTTF of Weibull Distribution =)=5000)=4431.13hr ,

where

* Simulational Explanation-

The Mean Time To Failure (MTTF) in reliability engineering is defined as the total expected operating time of a system before failure. It is given by the integral of the reliability function:

MTTF=

where R(t) is the probability that the system is still functioning at time t.

### Reliability Model (Weibull Distribution)

In this case, the reliability function is assumed to follow a Weibull form:

R(t) = exp [ − ( t / θ )ᵝ ]

where:  
θ = scale parameter  
β = shape parameter

So the MTTF becomes: MTTF=

Since integrating to infinity is not practical for numerical methods, the upper limit is replaced by a sufficiently large value tₘₐₓ:

MTTF ≈ ∫₀ᵗᵐᵃˣ exp [ − ( t / θ )ᵝ ] dt

tₘₐₓ is chosen such that R(tₘₐₓ) ≈ 0.

### Variable Transformation

To apply Monte Carlo integration, the variable is normalized:

x = t / tₘₐₓ  
t = x · tₘₐₓ  
dt = tₘₐₓ dx

Substituting:

MTTF ≈ tₘₐₓ dx

### Monte Carlo Estimation

Let xᵢ be uniformly distributed random numbers in [0,1]. Then:

MTTF ≈ ( tₘₐₓ / N ) Σ exp [ − ( xᵢ tₘₐₓ / θ )ᵝ ]  
for i = 1 to N

**5.2. Algorithm:**

**STEP 1:** Initialize sum = 0

**STEP 2:** FOR i = 1 to N DO

  2.1 Generate a random number x uniformly in the range [0,1]

  2.2 Compute reliability value:  
   y = exp [ − ( (x × t\_max) / theta ) ^ beta ]

  2.3 Update sum:  
   sum = sum + y

**STEP 3:** Compute Monte Carlo MTTF:

mttf\_mc = (t\_max / N) × sum

**STEP 4:** Compute percentage error:

error = | (mttf\_mc − mttf\_exact) / mttf\_exact | × 100

**STEP 5:** Display mttf\_mc and error

**STEP 6:** END

**5.3. Result:** Analytical MTTF: 4431.13

Experimental MTTF: 4430.94

Percentage Error: 0.0045% when tmax=11700hr

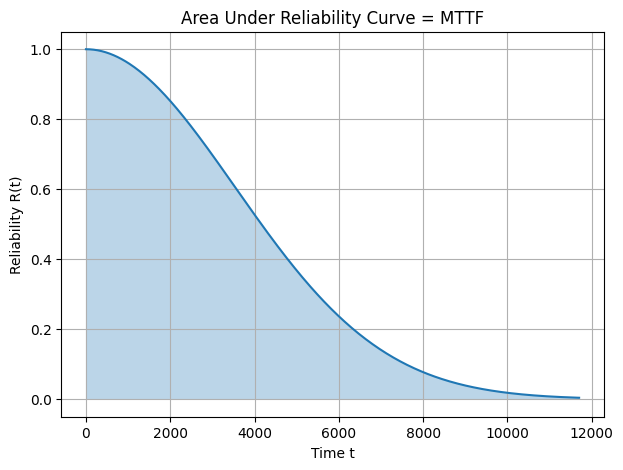


Fig 5.3.1. Graphical representation of Reliability vs Time

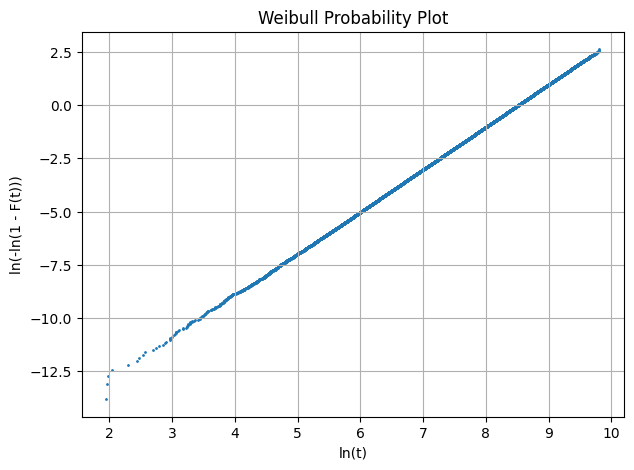


Fig 5.3.2: Weibull Probability Plot

## **5.4.OBSERVATIONS**

1. The Weibull probability plot shows the data points forming an approximately straight line.
2. This linear pattern confirms that the failure times follow a Weibull distribution.
3. The slope of the line is positive, indicating a shape parameter β > 1, which represents wear-out type failures.
4. The reliability curve starts at R(0) = 1 and decreases smoothly toward zero as time increases.
5. The shaded region under the reliability curve represents the Mean Time To Failure (MTTF).
6. The curve gradually approaches zero near t = t\_max, confirming that the selected upper limit is sufficient for numerical integration.

## **5.5. DISCUSSION**

The Weibull probability plot forms an approximately straight line, confirming that the failure data follows a Weibull distribution and validating the reliability model used. The reliability curve shows a gradual decrease from 1 to 0, representing increasing failure probability over time. The shaded area under this curve corresponds to the Mean Time To Failure (MTTF), which is estimated numerically using Monte Carlo sampling. The close agreement between analytical and simulated MTTF values demonstrates the accuracy and effectiveness of the Monte Carlo method for reliability analysis.

## **5.6.CONCLUSION**

The Monte Carlo method successfully estimates the Mean Time To Failure for a Weibull reliability model. The Weibull probability plot confirms that the failure data follows a Weibull distribution, validating the use of the reliability function.

The reliability curve clearly illustrates that MTTF is equal to the area under the survival curve. The numerical results closely match the analytical MTTF obtained using the Gamma function, demonstrating the accuracy of the simulation approach.

Therefore, Monte Carlo integration is a reliable and efficient technique for MTTF estimation and can be applied to complex reliability models where closed-form solutions do not exist.

**Problem\_6:** Integration - Estimate the value of the definite integral ∫₀¹ x³ dx

using:

1. Analytical method
2. Monte Carlo simulation method

* ANALYTICAL METHODOLOGY

The given function is:

f(x) = x³

The definite integral is evaluated analytically as follows:

∫₀¹ x³ dx

= [ x⁴ / 4 ]₀¹

= (1⁴ / 4) − (0⁴ / 4)

= 1/4

= 0.25

Analytical Result = 0.25

* SIMULATION METHODOLOGY –
* Generate N random numbers uniformly between 0 and 1
* Evaluate the function f(x) = x³ at each random value
* Compute the average of the function values
* Multiply the average by the interval length (b − a) = 1

Integral ≈ (b − a)/N × Σ f(xᵢ)

**6.1.Algorithm:**

**STEP 1:** Set total = 0

**STEP 2:** FOR i = 1 to N DO

  2.1 Generate a random number x uniformly in the interval [a, b]

  2.2 Evaluate the function value: f = x³

  2.3 Update total: total = total + f

**STEP 3:** Compute the Monte Carlo integral: I = (b − a)/N × total

**STEP 4:** Display I

**STEP 5:** END

TRIALS (N) f(x)=x³ Monte Carlo Integral

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

100 0.005635 0.298066

1000 0.007393 0.239595

10000 0.205356 0.254954

100000 0.402776 0.248405

Fig 6.1.1: Table of Simulation Approach

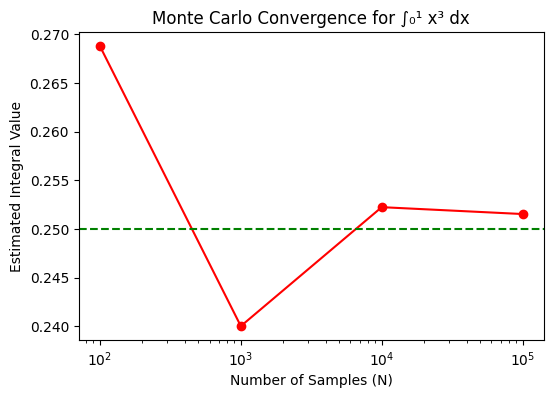
****

Fig 6.1.2: Graphical representation of Estimated Integral Value vs Number of Samples

## **6.2.OBSERVATION**

1. The Monte Carlo estimated values fluctuate for small sample sizes.
2. As the number of samples (N) increases, the estimated integral moves closer to the true value (0.25).
3. The plotted points gradually converge toward the horizontal dashed line representing the analytical solution.
4. The variation in results decreases with increasing N, indicating improved accuracy.

## **6.3.DISCUSSION**

The convergence plot illustrates the behavior of the Monte Carlo integration method for estimating the integral of x³ over the interval [0,1]. For small values of N, the estimates show noticeable deviation due to random sampling effects. However, as the number of samples increases, the computed values approach the analytical solution of 0.25. This demonstrates the law of large numbers, where increasing random samples reduces statistical error. The horizontal true-value line provides a clear reference showing how the numerical results converge toward the exact integral value.

## **6.4.CONCLUSION**

The Monte Carlo method effectively approximates the definite integral ∫₀¹ x³ dx. Although the results fluctuate for smaller sample sizes, increasing the number of trials significantly improves accuracy. The convergence toward the analytical value confirms the reliability of Monte Carlo simulation for numerical integration, especially for problems where analytical solutions may be difficult or unavailable.

**CONCLUSION**

Monte Carlo simulation is a powerful tool for analysing probabilistic systems such as coin tosses, dice rolls, and reliability models. In addition to estimating event probabilities, it is widely used to study series and parallel system reliability, capacity versus demand behaviour, system integration, and performance measures such as Mean Time to Failure (MTTF). This report demonstrates simulation-based probability estimation, convergence analysis, and error evaluation to validate theoretical concepts.

### Reference: 1) **Reliability Engineering Author:** Elsayed A. — Mohamed A. Elsayed

### 2)**Reliability and Life Testing Handbook Authors:** Dimitri Kececioglu

### 4) **Applied Reliability Authors:** John D. Andrews & Walter R. Moss

### 5) **Simulation Modeling and Analysis Authors:** A. M. Law & W. D. Kelton

### 7) **An Introduction to Reliability and Maintainability Engineering Author:** Charles E. Ebeling

### 8 ) NIST/SEMATECH e-Handbook of Statistical Methods• Covers probability

### distributions, reliability concepts <https://www.itl.nist.gov/div898/handbook/>

### 9) MathWorks Reliability & Monte Carlo Tutorials Applied simulation examples <https://www.mathworks.com>