PROJECT - 2 MONTE CARLO METHODS

NAME - Tanoghna Chattopadhyay
STUD. ID - 85 4132 4935

PROBLEM STATEMENTS

QUES 1 : Estimate π by the area method including confidence intervals on your estimate. Draw a graph of the successive values of the estimator as the number of samples increases. How many points do you need to use for your estimate to be within $\pm 1\%$ of the true value of π (with probability 0.95)?

QUES 2 : Consider a deck of cards (for simplicity numbered $1, \cdots, N$) Use a uniform random number generator to pick a card and record what card it is (if you were using actual cards, you would replace the card back into the deck - that is not necessary here as we never really take the card out of the deck). Repeat this N times, recording the number of times that each of the cards is selected. Some cards will not show up and some will show up more than once. Use this data to estimate

$$p_j = Pr\{\text{a card will be selected } j \text{ times in } N \text{ selections}\}$$

Run for $N = 10, 52, 100, 1000, 10000$ and verify $p_0 = 1/e$. Can you also find values for other p_j based on a mathematical analysis?

QUES 3 : Find $\hat{y}$, an estimate for Y and find a 95% confidence interval for the value of the integral

$$Y = \int_0^{\pi} \frac{\sin(x)}{x}\, dx.$$

Question 1: Es
1.1 Intro

THEORETICAL ANALYSIS

Monte Carlo simulation is a computerized mathematical technique that allows us to account for risk in quantitative analysis and decision making. The technique is used by professionals in widely disparate fields like finance, project management, energy, manufacturing, engineering and research.

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

They are used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables.

Question 1: Estimation of value of pi by area method

1.1 Introduction

To estimate the value of pi, a large sequence of uniformly distributed random numbers are generated. These numbers can take any value between 0 and 1. Now, if we consider a quarter circle of radius 1 unit inscribed in a square of length 1 unit, then, the area of the quarter of the circle will be (π*0.25) sq. unit and the area of the quarter of square will be 0.25 sq. unit.

This implies that if we pick M random points inside the square, N*π/4 of those points will lie inside the circle. By using the 'rand' function in MATLAB, M random points were generated and the location of the points(those lying inside and those lying outside the circle) were determined. The value of π is then given by the estimate of 4*M/N. The probability of the point lying inside the circle is given by N/M.

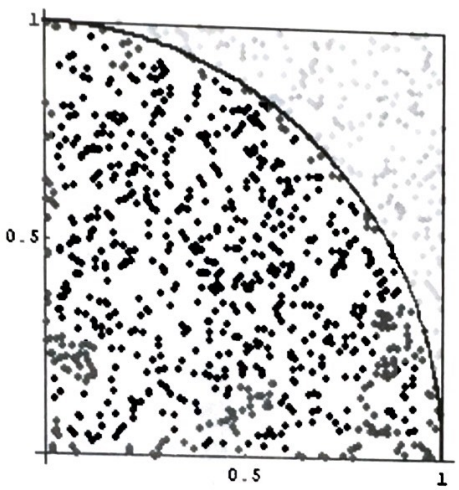


Fig : pi estimation by area method

1.2 Simulation Methodology

While simulating, we take 'n' points from the user. 'n' (generally a large value for a good estimate) is used to generate N points that will lie in the square of side 1 unit. The rand function in MATLAB will generate uniformly distributed x and y coordinates of the N points. After the points are generated, the program checks whether the points lie within the circle or not by verifying $x^2 + y^2 \leq 1$. If this condition is satisfied, the point lies within the quarter circle and hence the count for M, which indicates the number of points inside the circle, is incremented. If the condition is not satisfied, no change is made to the count M, indicating that the point does not lie inside the circle. After reiterating for n values, using M and N, the value of π is estimated by 4*M/N. This estimated value is very close to the actual value.

A confidence interval(CI) is a type of interval estimate (of a population parameter) that is computed from the observed data. These intervals contain a range of values that act as a close estimate of some parameter. In the program, the confidence interval is considered to be 95%. The confidence intervals in the program give a range of estimated values of π.

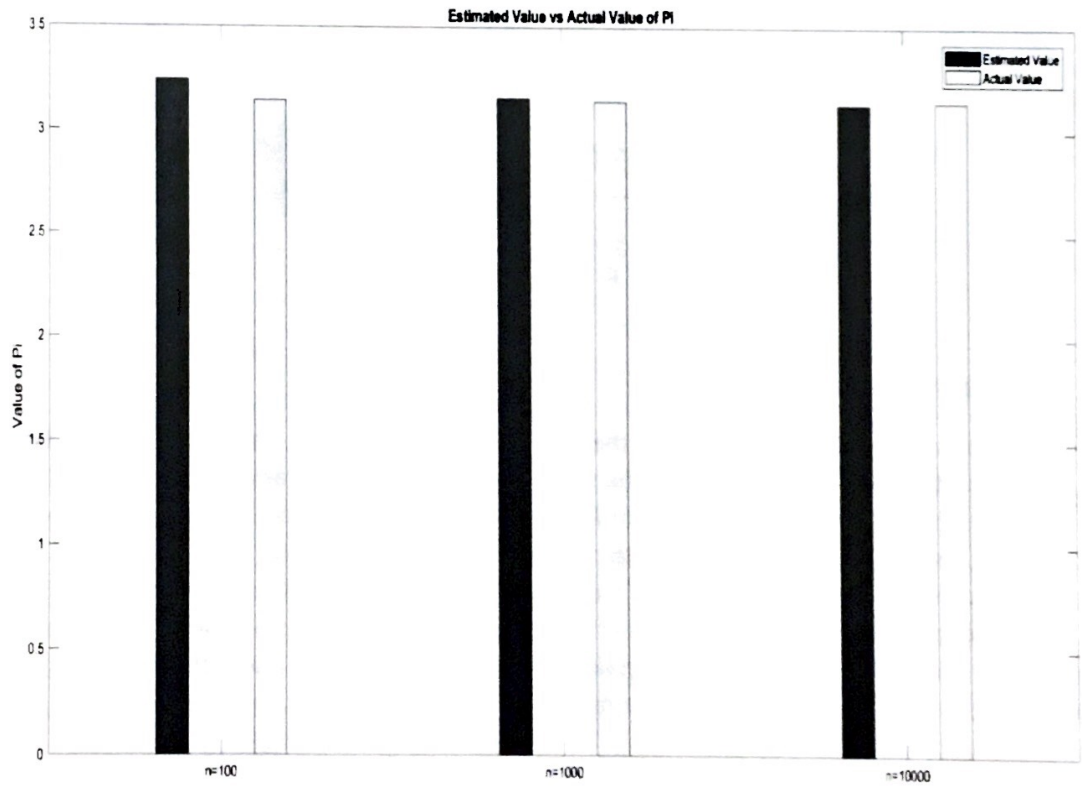
The mean of the values of π and the standard deviation for the area is computed. After this, the margin of error for a 95% confidence interval is evaluated. Using the 'tinv' function, estimated pi and the error margin, the confidence interval is generated.

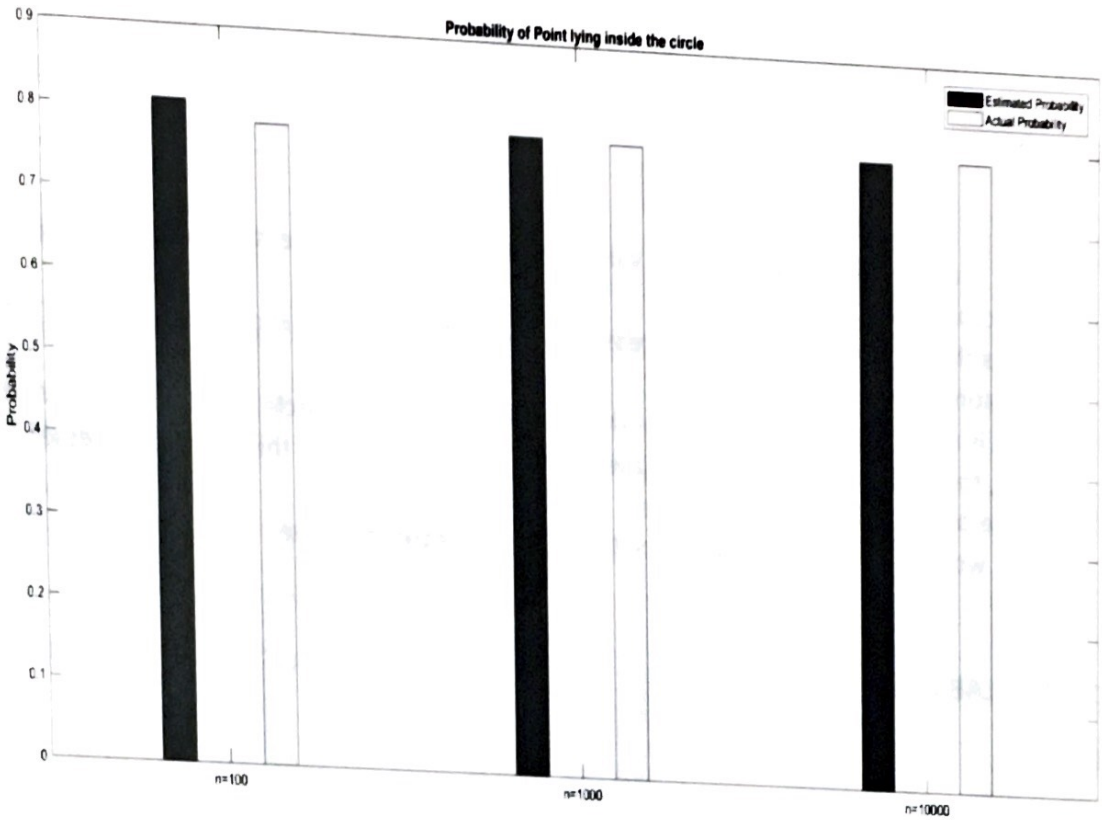
For 3 different values of n, the value of π is estimated. After generating the three values, a comparison graph with these values is plotted. The probability P of the point lying within the circle is also determined and comparison is also done for the three cases. The program also determines whether the large value of n are sufficient enough or not to estimate the value of π within 1% of the actual value. It checks whether the value of π estimated lies within the range. If it does not, it gives feedback that more values of n will be required to give a better estimate. It also plots the continuous plot of estimated value of pi for 100000 samples.

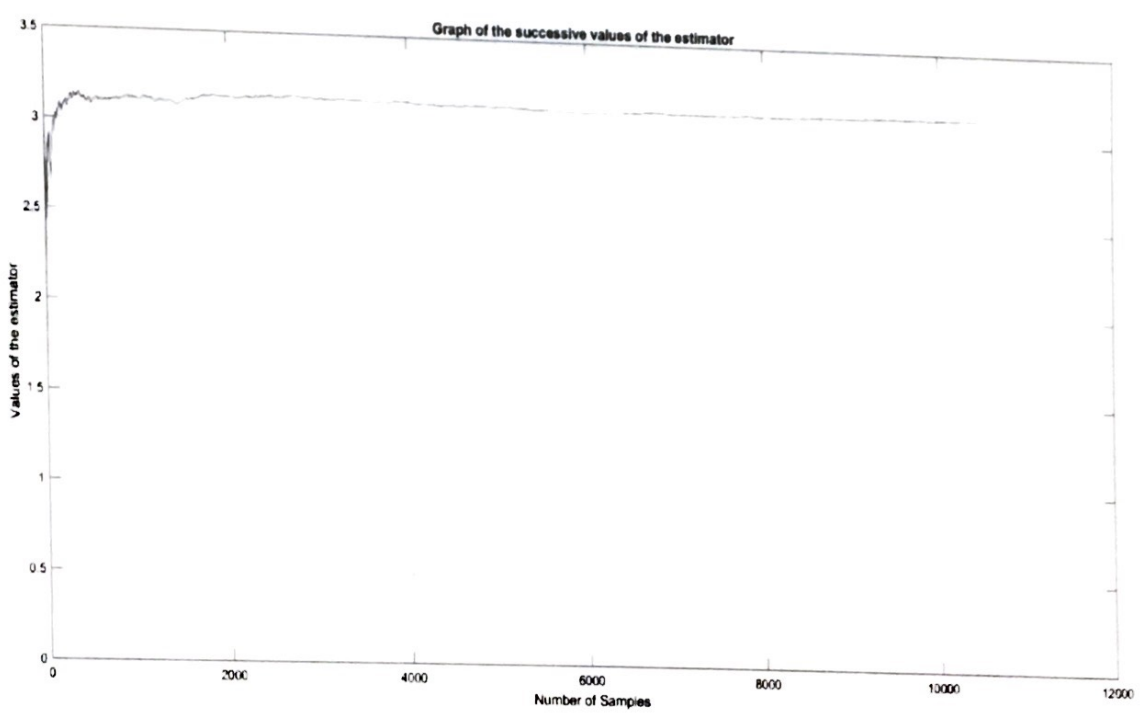
1.3 Results

So for the various n, the estimates are:

Value of n	100	1000	10000
Estimated value of π	3.2400	3.1560	3.1288




Probability of Point lying inside the circle
Estimated Probability
Actual Probability
Probability
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
n=100
n=1000
n=10000


Graph of the successive values of the estimator
Values of the estimator
3.5
3
2.5
2
1.5
1
0.5
0
0
2000
4000
6000
8000
10000
12000
Number of Samples

Even at n = 10000, it says that more values are required. The required values are 1.0108*10^4.

1.4 Observations and Inferences

- It is observed that with a larger value of n, the estimated value is approximately equal to the actual value of pi.
- As the number of iterations increase, the closeness to the value of pi increases for the estimation.
- It is noted that the probability of the point lying within the circle is π/4= 0.78539. With a larger n, the probability of a point lying within the circle approaches this value. With lesser number of iterations, this probability varies.
- It would be better to increase n>10000 to get a better estimate.

1.5 MATLAB Code

Part A

```
% Estimation of pi by area method

% no. of simulations
num = 100;
N = 0;
M = 0;

% Generate uniform distributed random variables
for i = 1:num
    x = rand(2,1);
    A(i) = x(1)^2 + x(2)^2;

    % If point lies within circle, then increment value
    if(x(1)^2 + x(2)^2 <=1)
        N = N+1;
        M = M+1;
    else(x(1)^2 + x(2)^2 >1)
        N = N+1;
    end
end

% Estimated value of pi
A_pi1 = 4*M/N;

% Probability of point lying inside the circle
P1 = M/N;

S = std(A)/sqrt(num);
T = tinv([0.025  0.975],num-1);

% Confidence interval for pi
```

```
CI1 = mean(A_pi1)+ T*S;

% Confidence interval for Probability
CI2 = mean(P1)+- T * (S/ square(num));

pbar = pi/4;

% Mean of pcap
xbarpbar = pi/4;

% Variance of pcap
varpbar = (pi/4)*(1-(pi/4));

% Standard deviation of pcap
stdpbar = sqrt(varpbar);

% For 95% confidence
b = 1.95;

% 1% error in value of pi/4
e = 0.01;
n = (b*stdpbar/(e*pbar))^2;

% no. of simulations
num = 1000;
N = 0;
M = 0;

% Generate uniform distributed random variables
for i = 1:num
    x = rand(2,1);
    A(i) = x(1)^2 + x(2)^2;

    % If point lies within circle, then increment value
    if(x(1)^2 + x(2)^2 <=1)
        N = N+1;
        M = M+1;
    else(x(1)^2 + x(2)^2 >1)
        N = N+1;
    end
end

% Estimated value of pi
A_pi2 = 4*M/N;

% Probability of point lying inside the circle
P2 = M/N;

S = std(A)/sqrt(num);
T = tinv([0.025  0.975],num-1);

% Confidence interval for pi
CI1 = mean(A_pi2)+ T*S;

% Confidence interval for Probability
CI2 = mean(P2)+- T * (S/ square(num));
```

```
pbar = pi/4;

% Mean of pcap
xbarpbar = pi/4;

% Variance of pcap
varpbar = (pi/4)*(1-(pi/4));

% Standard deviation of pcap
stdpbar = sqrt(varpbar);

% For 95% confidence
b = 1.95;

% 1% error in value of pi/4
e = 0.01;
n = (b*stdpbar/(e*pbar))^2;

% no. of simulations
num = 10000;
N = 0;
M = 0;

% Generate uniform distributed random variables
for i = 1:num
    x = rand(2,1);
    A(i) = x(1)^2 + x(2)^2;

    % If point lies within circle, then increment value
    if(x(1)^2 + x(2)^2 <=1)
        N = N+1;
        M = M+1;
    else(x(1)^2 + x(2)^2 >1)
        N = N+1;
    end
end

% Estimated value of pi
A_pi3 = 4*M/N;

% Probability of point lying inside the circle
P3 = M/N;

S = std(A)/sqrt(num);
T = tinv([0.025  0.975],num-1);

% Confidence interval for pi
CI1 = mean(A_pi3)+ T*S;

% Confidence interval for Probability
CI2 = mean(P3)+- T * (S/ square(num));

pbar = pi/4;

% Mean of pcap
xbarpbar = pi/4;
```

```
% Variance of pcap
varpbar = (pi/4)*(1-(pi/4));

% Standard deviation of pcap
stdpbar = sqrt(varpbar);

% For 95% confidence
b = 1.95;

% 1% error in value of pi/4
e = 0.01;
n = (b*stdpbar/(e*pbar))^2;

y = [A_pi1 pi; A_pi2 pi; A_pi3 pi];
j = [P1 pi/4; P2 pi/4; P3 pi/4];
bar(y, 0.33)
title('Estimated Value vs Actual Value of Pi');
ylabel('Value of Pi');
legend('Estimated Value','Actual Value');
set(gca,'XTick',1:3,'XTickLabel',{'n=100','n=1000','n=10000'});

figure;
bar(j, 0.33)
title('Probability of Point lying inside the circle');
ylabel('Probability');
legend('Estimated Probability','Actual Probability');
set(gca,'XTick',1:3,'XTickLabel',{'n=100','n=1000','n=10000'});

% used to determine if the estimate is in the range of +- 1% of value of pi.
if (((pi-0.01*pi)<=A_pi3) | (A_pi3<=(pi+0.01*pi)))
    disp('more values required');
else
    disp('n1 values required');
end
```

Part B

```
% Graph of successive values of estimator for value of pi

clc;
n_sample = 100000;
X = rand(1,n_sample);
Y = rand(1,n_sample);
P = zeros(1,n_sample);
Pi_pest = zeros(1,n_sample);
p_est = zeros(1,n_sample);
s_pbar = 1;
i = 1;

while s_pbar>0.01*pi*0.25/1.96
    i = i+1;
    if ((X(i)^2 + Y(i)^2) <= 1)
        P(i) = 1;
    end
    p_est(i) = sum(P(1:i))/i;
    Pi_pest(i) = (P(i) - p_est(i))^2;
```

```
    s_pbar = sqrt(sum(Pi_pest(1:i))/(i*(i-1)));
end

x = 1:i;
y = 4*p_est(1:i);
plot(x,y)
grid on;
title('Graph of the successive values of the estimator');
xlabel('Number of Samples');
ylabel('Values of the estimator');
n_points = i
Pi_est = 4*p_est(i)
```

QUESTION 2 : Estimation of value of Euler's Number

2.1 INTRODUCTION

If we consider $\{ X_i : i = 1, \cdots, n \}$ a series of samples from a uniform distribution over the integers $\{1, 2, \cdots, n\}$. Then the probability of a number not appearing in the series is given by,

$$\begin{aligned} Pr\{\text{No num } j\} &= Pr\{ X_1 \neq j, X_2 \neq j, \cdots X_n \neq j \} \\ &= \prod_{i=1}^{n} Pr\{ X_i \neq j \} \\ &= \prod_{i=1}^{n} \left[1 - \frac{1}{n} \right] \\ &= \left(1 - \frac{1}{n} \right)^n \end{aligned}$$

Now, we know,

$$\exp(-1) = \lim_{n \to \infty} \left(1 - \frac{1}{n} \right)^n$$

so, we can write,

$$\exp(-1) \approx \hat{p} = \lim_{n \to \infty} \left(1 - \frac{1}{n} \right)^n$$

$$\therefore \lim_{n \to \infty} \left(1 - \frac{1}{n} \right)^n = \frac{1}{e} = 0.368$$

Thus, our estimate of p_r is found by estimating the probability that a specific observation is omitted from n draws.

2.2 SIMULATION METHODOLOGY

While simulating, we take 'n' trials (generally a large value for a good estimate). We then generate a fixed random number 'k' to depict the card which we look for in each draw. We also assign a counter.

Then running a loop for the 'n' trials, we set a flag as false to indicate not seeing the number 'k' and count 1 = 1. Then while the value of count 1 is less than n and flag is false, we generate a random integer using the 'randi' function called 'r'. This 'r' is checked whether equal to 'k'. If they are equal, then flag is set to 1. The count 1 is incremented by 1.

At the end of the loop, if flag is still false, then we increment the value of count by 1. The probability of the card being selected is the ratio:

$$p = \frac{count}{n}$$

The value of this probability indicates the number of times 'k' appeared in n draws.

The value p should be approximately equal to 1/e. This is because we expect that 37% of ~~time~~ time, the card 'k' will not be selected.

2.3 RESULTS

Value of N	P_0
10	0.3000
52	0.3654
100	0.3900
1000	0.3470
10000	0.3710

We know the value of $1/e$ = 0.3678

2.4 OBSERVATIONS AND INFERENCES

- It is observed that P_0, i.e., P_H(a card will be selected 0 times in N selections) is approximately equal to $1/e$.
- As we increase the value of N, the value of P_r gets closer to the value of $1/e$.
- P_r (no num j) $= P_r \{ x_1 \neq j, x_2 \neq j, \cdots, x_n \neq j \}$

$$= \prod_{i=1}^{n} P_r \{ x_i \neq j \}$$

$$= \prod_{i=1}^{n} \left[1 - \frac{1}{n} \right]$$

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

And,

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} = 0.368$$

- P_r (a card is selected j times in N selections)

$$= P_r \{ X_1 = k, \cdots X_j = k, X_{j+1} \neq k, \cdots X_n \neq k \}$$

$$= \prod_{i=1}^{j} P_r \{ X_i = k \} \cdot \prod_{i=j+1}^{n} P_r \{ X_i \neq k \}$$

$$= \prod_{i=1}^{j} \left(\frac{1}{n}\right) \prod_{i=j+1}^{n} \left(1 - \frac{1}{n}\right)$$

This is the mathematical analysis behind P_n.

- Pseudo code :

```
repeat sum number of time (number of trials)
    set a flag to false to indicate you have not seen number k
    loop counter = 1
        while (loop counter <= n) and (flag is false)
            pick a number between 1 and n
            if it is equal to k set the flag to true
            counter = counter + 1
            end loop
    if flag is false (i.e., we didn't find k) increment the
    count of the number of times this happened
end number of trials loop.
```

2.5 MATLAB CODE

```
% Question 2

% no. of simulations
num = 10000;

% Random number to be searched for
k = randi([1 num], 1);
count = 0;
for i = 1:num
    flag = 0;                          % To indicate Not Seeing the number
    count1 = 1;
    while(count1<=num && flag==0)
        r = randi([1 num], 1);
        if(r == k)
            flag = 1;
        end
        count1 = count1+1;
    end
    if(flag == 0)
        count = count+1;
    end
end

p = count/num;                         % Probability value
```

QUESTION 3 : Evaluation of Integral

3.1 INTRODUCTION

Evaluation of integrals using monte Carlo methods can be done using uniformly distributed number between 0 and 1, i.e., Uniform (0, 1), the strong law of large numbers and the central limit theorem.

For evaluating $I = \int_0^1 f(x)\,dx$, we can express I as $I = E[f(U)]$ where U is uniformly distributed over 0 and 1. If $U_1, U_2, \cdots, U_n$ are independent uniform random variables, it follows that the uniform random variables $f(U_1), f(U_2), \cdots, f(U_n)$ are independent and identically distributed random variables having a mean of I. Using strong law of large numbers, with probability 1, we get,

$$\sum_{i=1}^{n} f(U_i)\,\frac{1}{n} \longrightarrow E[f(U)] = I \quad \text{as } n \longrightarrow \infty$$

Thus, I can be approximated using a large number of random number U_i and taking the average of the generated $f(U_i)$.

To evaluate an integral with limits (a, b) instead of (0, 1) we need to use certain substitutions to get the integral in the form $\int_0^1 f(x)\,dx$.

$$I = \int_a^b f(x)\,dx$$

Let $y = \frac{x-a}{b-a}$, then, $dy = \frac{dx}{b-a}$

So,

$$I = \int_0^1 f(a + [b-a]\,y)\,(b-a)\,dy$$

$$= \int_0^1 g(y)\,dy$$

where,

$$g(y) = (b-a)\,f(a + [b-a]\,y)$$

Further, to get the approximate value of the integral, the average values of the sum is taken into consideration:

$$I = \left(\frac{b-a}{n}\right) \cdot \sum_{i=1}^{n} f(x_i)$$

3.2 SIMULATION METHODOLOGY

A large uniformly distributed sequence of numbers (z) between 0 and 1 is taken as input. Using the 'int' function on MATLAB, the value of the integral of $\sin(x)/x$ is evaluated over the interval $((n-1)\pi, n\pi)$. For different integer values of 'n' (1, 2, 3, 4, 5), the integral is evaluated.

The limits of the sum for the estimation are set by the value of n, $a = (n-1)\pi$ and $b = n\pi$. Reiterating or summing the values estimated of $\sin(x)/(x)$ 'z' times and taking its average, gives the estimated value of the integral. With the increasing values of n and due to the nature of the sin function, the integral can also be estimated for $n = 2, 3, 4, 5$ using the values of 'z' for $n = 1$.

Using this technique we can evaluate

$$D(n) = \int_0^{n\pi} \frac{\sin(x)}{x}\, dx$$

We have used, $a = 0$ and $b = n\pi$ for $n = 1, 10, 100, 1000$

3.3 RESULTS

All the results are for $z = 10000$ random numbers

n	Expected Value	Approx Value	Error
10	1·5390	1·5527	-0·8902 %
100	1·5676	1·6384	-4·5164 %
1000	1·5924	1·5705	- 1·3945 %

For $n = 1$, and $z = 10000$
we get

the approximate value

$$\int_0^{\pi} \frac{\sin(x)}{x}\, dx = 1·8559$$

The bar graph for the comparison of the above table is attached with the code.

Confidence Interval : 1·8388 1·8743

3.4 OBSERVATIONS AND INFERENCES

- It is observed that with increasing the number of random variables / iterations the estimated value of the integral is very close to the expected value of the integral

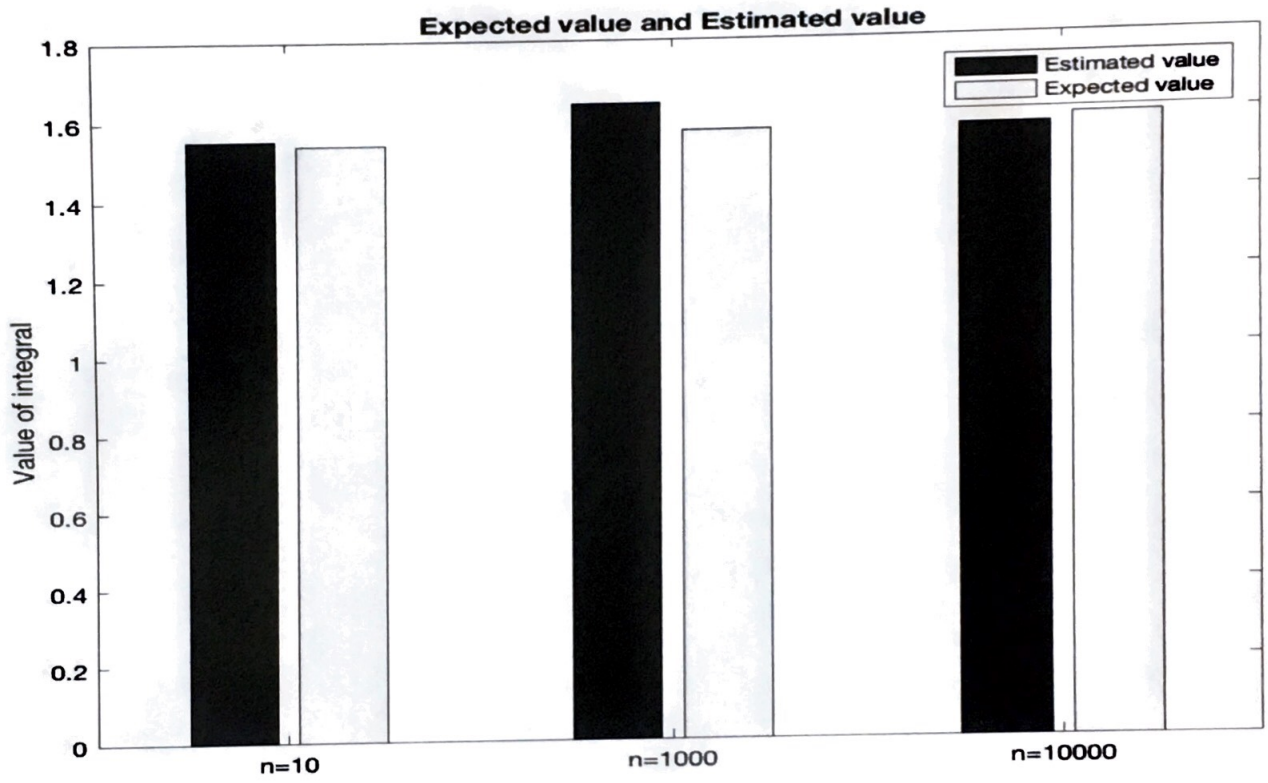
- $$\int_0^{\infty} \frac{\sin(x)}{x}\, dx = \frac{\pi}{2}$$

Inserting the correct values for upper and lower limit in the code gives value $\frac{\pi}{2}$

- $$\int_0^{\pi} \frac{\sin(x)}{x}\, dx = 1.8559$$

- The error observed is very small and hence this method gives approximately the same answer as obtained by evaluation of the integral.

Estimated vs Expected Value of the integral



3.5 MATLAB CODE

```
% Monte Carlo approach for evaluation of integrals.

clear all

n = input('Enter value for n: ')
syms x;

% Expected value of integral
Ic = int(sin(x)/(x),0,n*pi)

% Random number generator
z = input('Enter z for a large sequence: ')

% Limits of integrals
a = (n-1)*pi;
b = n*pi;
x =(b-a)*rand(1,z)+a;
for i=1:z
    fx(i)=sin(x(i))/(x(i));  %sin(x)/(x)
end

% Average of sum to get approximate value
Imc =(b-a)*(sum(fx)/z)

% Error
error = 100*(Ic-Imc)/Ic

% Confidence Interval
S = std(x)/sqrt(z);
T = tinv([0.025  0.975],z-1);
CT = mean(Imc) + T*S;
```