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.

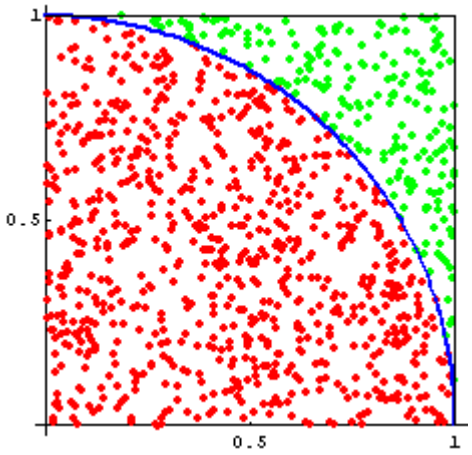


Fig : pi estimation by area method

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.

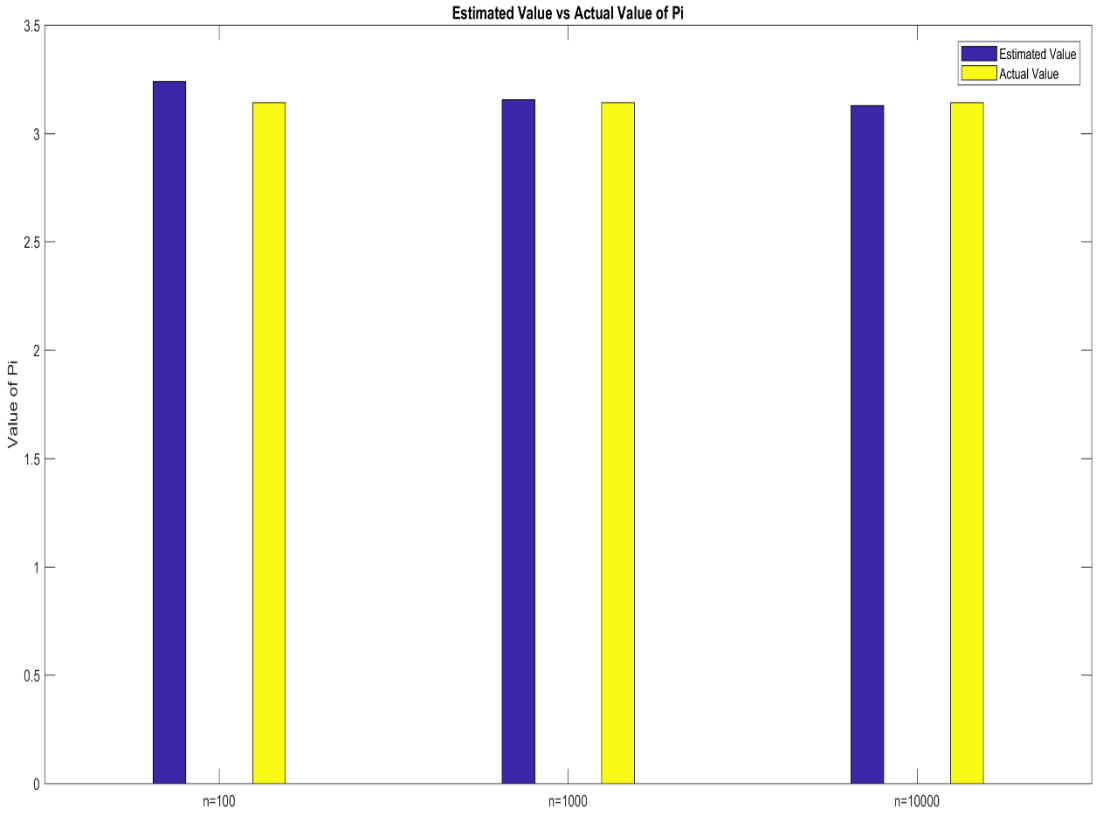
The mean of the values of π and the standard deviation for the area is computed. After this, the margin of error for a 95% confidence interval is evaluated. Using the 'tinv' function, estimated pi and the error margin, the confidence interval is generated.

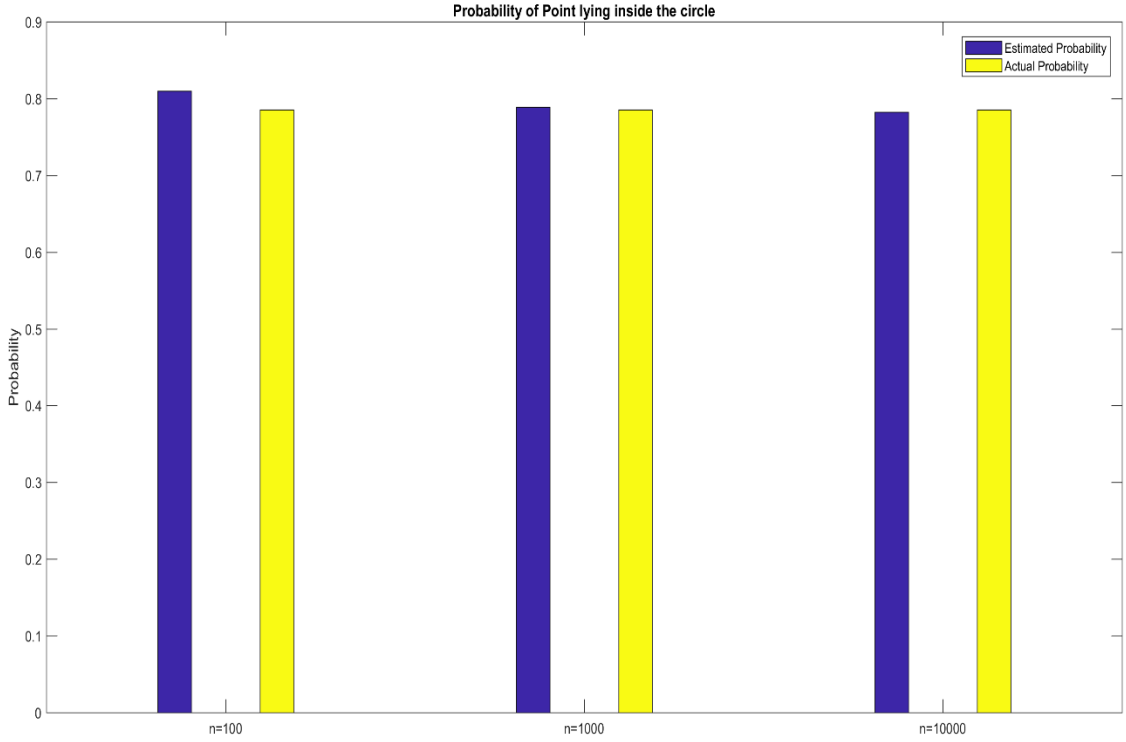
For 3 different values of n, the value of π is estimated. After generating the three values, a comparison graph with these values is plotted. The probability P of the point lying within the circle is also determined and comparison is also done for the three cases. The program also determines whether 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.

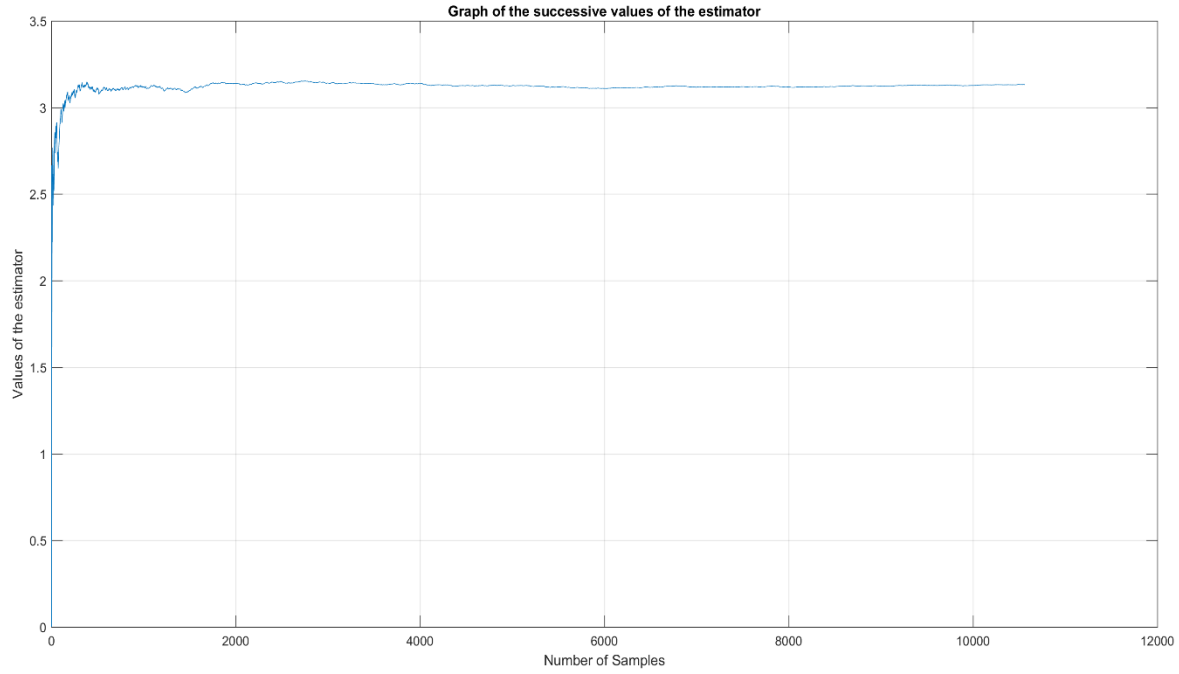
1.3 Results

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)
```