Student name:	
Student name:	

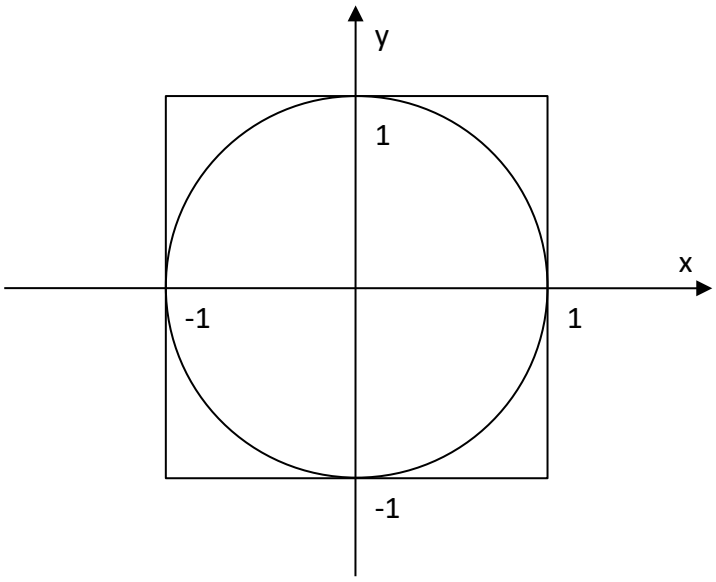
MATLAB, Lab 10 – group work

Monte Carlo Simulation

Monte Carlo methods are class of simulations based on randomly generated numbers. Its idea was created by a mathematician Stanisław Ulam while playing Solitaire. He tried to find a way of evaluating whether the Solitaire is going to be successful based on the set of 52 cards. After many attempts, he decided to change the approach and play games one after another in order to estimate the chances of winning by simply counting number of successful games. First simulations were applied within Manhattan project in Los Alamos together with great American mathematicians and physicists working on nuclear weapon (von Neumann, Feynman, Fermi, Bethe, Gamow). This method's validity emerges from principal probability statement saying that the ratio of successful hits will be approaching hitting probability when number of iterations is sufficiently large. These methods are popular in approaching problems having deep mathematical complexity, or uncertain boundary conditions. This includes mainly:

- Fluid flows
- Crystal growth
- Business/economic risk assessments
- Predictions of currency conversion rates
- Multi-dimensional integrals

Serious MC simulation demands huge set of random numbers. That occurs to be the main problem with these algorithms. It is caused by the fact that all random number generators available at different codes are in fact "pseudo-random". Numbers are generated based on machine clock or another parameter changing in time. Therefore, it can occur that the space is not uniformly covered with the set of random points, especially for extremely large number of iterations.



ASSIGNMENT:

Create set of n_s random points contained within a square presented in the figure above. Afterwards, evaluate the number n_c of points lying within the circle. Assuming perfect uniformity of the generated random number set one can say that for sufficiently large n_s the following relation will be satisfied:

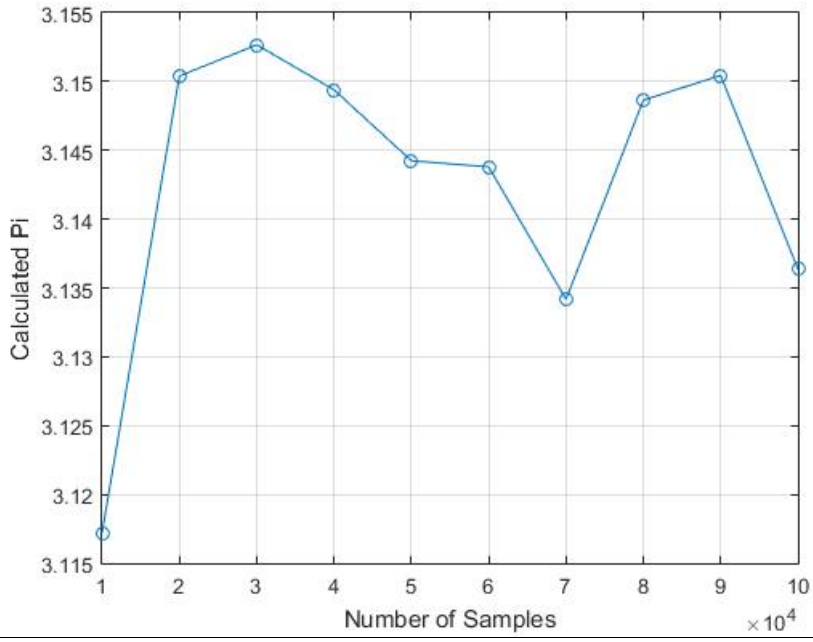
$$\frac{n_c}{n_s} = \frac{A_c}{A_s},$$

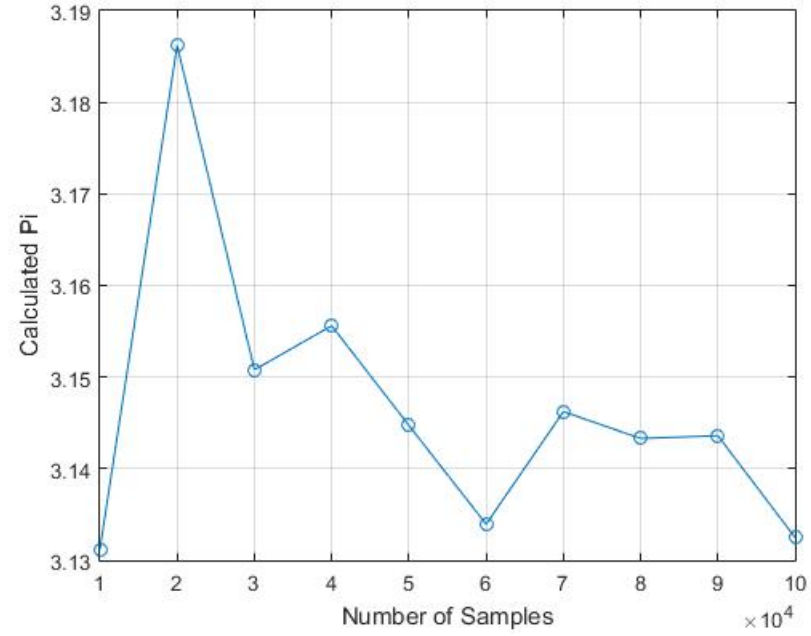
Where A_c, A_s are adequate areas: that of a circle and that of a square. Things to do:

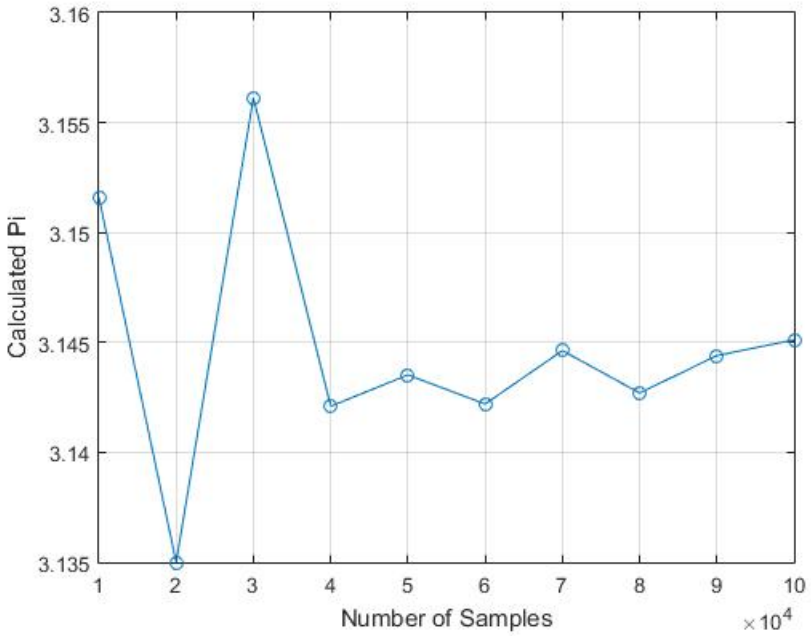
1. Repeat simulations for increasing n_s and plot calculated π value versus n_s. Paste the plot and the code into the field below. Comment on the amount of time necessary to reach the consecutive solution as you increase the "number of hits".

Code:

```
for t=1:10

   N=t*10000;              % Total Number Of Samples
   x_coordinate = 2*rand(1,N)-1;      % N samples between -1 and 1
   y_coordinate = 2*rand(1,N)-1;      % N samples between -1 and 1

   n=0;
    for i=1:N
       if(x_coordinate(i)^2+y_coordinate(i)^2 <=1)
          n=n+1; %Number Of Samples Inside The Circle
       end
    end

     pi(t) = (4*n)/N;  % Formula of Pi
end

  z=(1:1:10)*10000;
  plot(z,pi,'-o')
  grid on;
  xlabel('Number of Samples')
  ylabel('Calculated Pi')
```





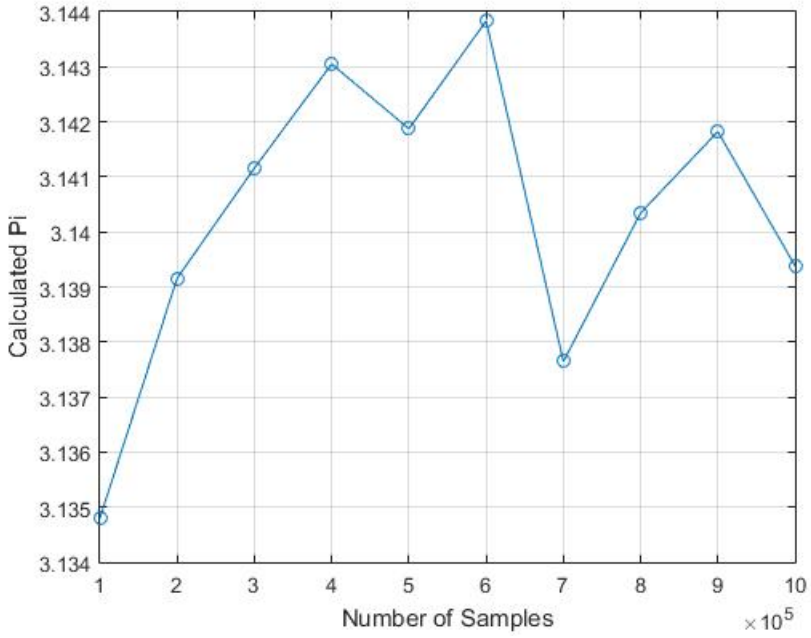


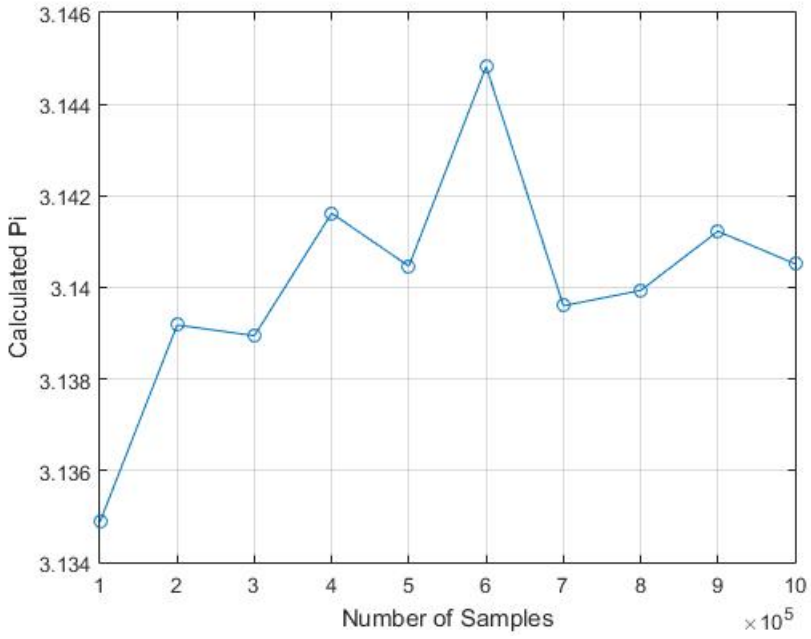
3 different generated random numbers vs calculated pi

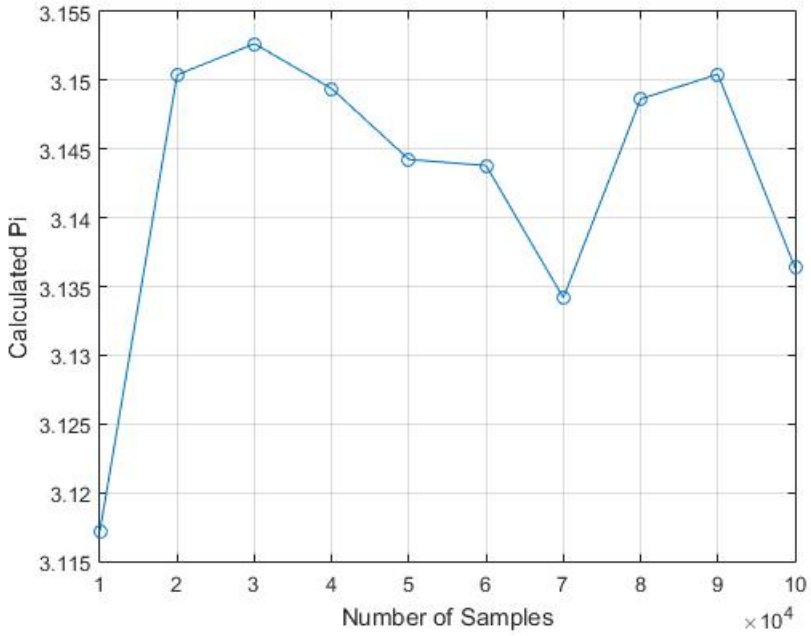
2. What is the value of n_s necessary to obtain π with the accuracy of 2 decimal numbers? Confirm your assumption with some arguments.

Answer:
Our accuracy in pi value depends on number of samples.The more examples we have mean more accurate results.In below,you can see pi value with the accuracy of 2 decimal

numbers.In this case,number of samples begin with 100.000 and increases 100.000 until it's 1.000.000.We can also obtain accuracy of 2 decimal numbers,samples between 10.000 and 100.000.






3.155
3.15
3.145
3.14
3.135
3.13
3.125
3.12
3.115
Calculated Pi
1
2
3
4
5
6
7
8
9
10
Number of Samples
×10^4