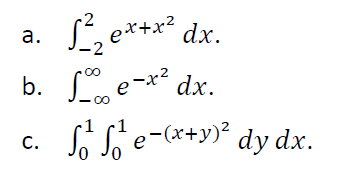
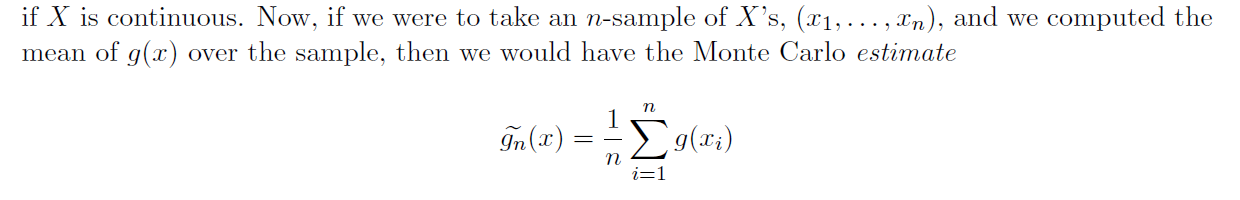
Project #4 – Integrals and Intervals

1. *Approximate the following integrals using a Monte Carlo simulation. Compare your estimates with the exact values (if known)*



**Concept behind this problem:**

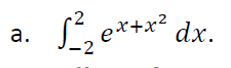
The question requires the knowledge of *Monte carlo simulation for Integrals.* Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values—a probability distribution—for any factor that has inherent uncertainty. It then calculates results over and over, each time using a different set of random values from the probability functions. Monte Carlo simulation produces distributions of possible outcome values. Monte Carlo simulation provides a number of advantages over deterministic, or “single-point estimate analysis:



**Code & Code Description:**

* Initialize variables and create an exponential function to calculate the theoretical value of the integral
* The function runs for 100, 1000, 10000 and 100,000 samples.

In the code, we perform the below for each case



* Generate a uniform random sample in the interval (0 1).
* Shift the random sample in the interval of (0 1) to (-2 2) by using

(b-a)\*rand(U) +b (a-upper bound, b-lower bound)

* Now take ‘n’ number of shifted samples like above and substitute it in the expr

(b-a)\*(exp(x+x^2).

* Repeat this experiment for number of runs mentioned above and finally calculate mean of the above.

function [ ] = monte\_carlo\_simulation()

%Initialize

exp\_fun = @(x) exp(x+x.^2);

Upper\_Bound = 2;

Lower\_bound = -2;

theo\_integral = integral(exp\_fun,Lower\_bound,Upper\_Bound);

disp(['Theoritical Integral = ',num2str(theo\_integral)]);

%Shift the (0,1) interval of integral to (-2,2)

runs = [1000 10000 100000];

for expt=1:3

cum\_sum = 0;

num\_of\_samples = runs(expt);

u\_samples = ((rand(1,num\_of\_samples))\*(Upper\_Bound-Lower\_bound))+ Lower\_bound;

for iter=1:num\_of\_samples

vector = ((Upper\_Bound-Lower\_bound)\*exp(u\_samples(iter) + (u\_samples(iter)\*u\_samples(iter))));

cum\_sum = cum\_sum + vector;

end;

disp(['Summary for ',num2str(num\_of\_samples)])

disp(['Calculated(via Monte Carlo) = ',num2str(cum\_sum/num\_of\_samples)]); end



* First we have to split the -inf to inf limit -inf to 0 and 0 to inf.
* Next we need to convert the interval of 0-inf to 0-1 for which we pick a y=1/(1+x) and perform the below
  + Get a uniform random number in (0 1).
  + Substitute the rand(U) in the expression taking the form where y=1/(1+x)
  + Calculate the same for -inf to 0 and 0 to inf and take the mean of both the summation.

function [ ] = monte\_carlo\_simulation()

%Initialize

exp\_fun = @(x) exp(-x.^2);

theo\_integral = integral(exp\_fun,-inf,inf);

disp(['Theoritical Integral = ',num2str(theo\_integral)]);

%Shift the (0,1) interval of integrl to (-2,2)

runs = [1000 10000 100000];

%Monte Carlo Simulation got Integrals

for expt=1:3

cum\_sum1 = 0;

cum\_sum2 = 0;

num\_of\_samples = runs(expt);

u\_samples = rand(num\_of\_samples,1);

%o to Inifinite Limit

for iter=1:num\_of\_samples

vector(iter) = ((1/u\_samples(iter))^2\*exp(-((1/u\_samples(iter)) -1)^2));

cum\_sum1 = cum\_sum1 + vector(iter);

end;

%-Inifinite to 0 Limit

for iter=1:num\_of\_samples

vector(iter) = ((1/u\_samples(iter)^2)\*exp(-((1/u\_samples(iter)) -1)^2));

cum\_sum2 = cum\_sum2 + vector(iter);

end;

disp(['Summary for ',num2str(num\_of\_samples)])

disp(['Calculated(via Monte Carlo) = ',num2str((cum\_sum1+cum\_sum2)/num\_of\_samples)]); end



* Here we have two variables ‘x’ and ‘y’. Therefore we first calculate the theo value of integral by forming integral with respect to ‘y’ followed by with respect to ‘x’
* We take two random variables X and Y in interval (0 1) and substitute in the expression.
* Repeat this experiment and calculate the mean of the value to get the integral value.

function [ ] = monte\_carlo\_simulation()

%Initialize

syms x y;

exp\_fun = exp(-((x+y)^2));

integral\_1 = int(exp(-((x+y)^2)),y,0,1);

theo\_integral = int(integral\_1,x,0,1);

sprintf('Theoritical Integral = %f', theo\_integral)

runs = [1000 10000 100000];

for expt=1:3

cum\_sum = 0;

num\_of\_samples = runs(expt);

X\_samples = rand(num\_of\_samples,1);

Y\_samples = rand(num\_of\_samples,1);

for iter=1:num\_of\_samples

vector(iter) = exp(-((X\_samples(iter)+Y\_samples(iter))^2));

cum\_sum = cum\_sum + vector(iter);

end;

disp(['Summary for ',num2str(num\_of\_samples)])

disp(['Calculated(via Monte Carlo) = ',num2str(cum\_sum/num\_of\_samples)]); end

**Results and Theoritical Values:**

**1A**

Theoritical Integral = 93.1628

Summary for 1000 Calculated(via Monte Carlo) = 88.6909

Summary for 10000 Calculated(via Monte Carlo) = 90.4676

Summary for 100000 Calculated(via Monte Carlo) = 93.1005

**1B**

Theoritical Integral = 1.7725

Summary for 1000 Calculated(via Monte Carlo) = 1.7739

Summary for 10000 Calculated(via Monte Carlo) = 1.7746

Summary for 100000 Calculated(via Monte Carlo) = 1.7765

**1C**

Theoritical Integral = 0.411793

Summary for 1000 Calculated(via Monte Carlo) = 0.41282

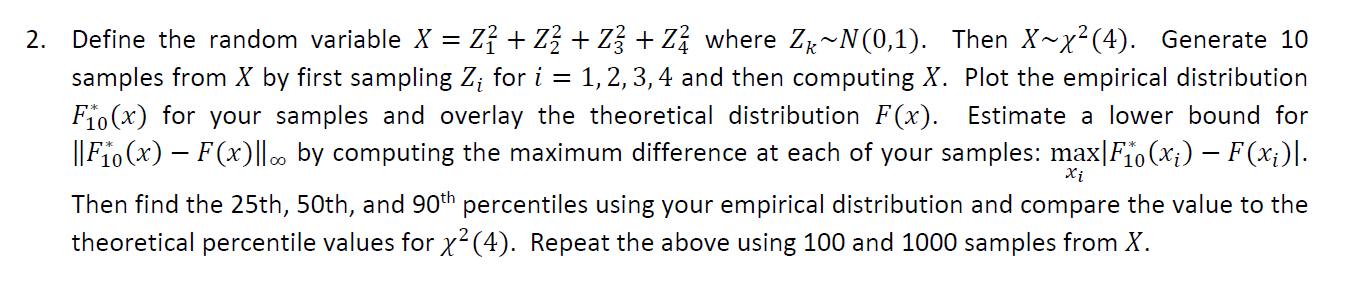
Summary for 10000 Calculated(via Monte Carlo) = 0.40757

Summary for 100000 Calculated(via Monte Carlo) = 0.41205

**Observations:**

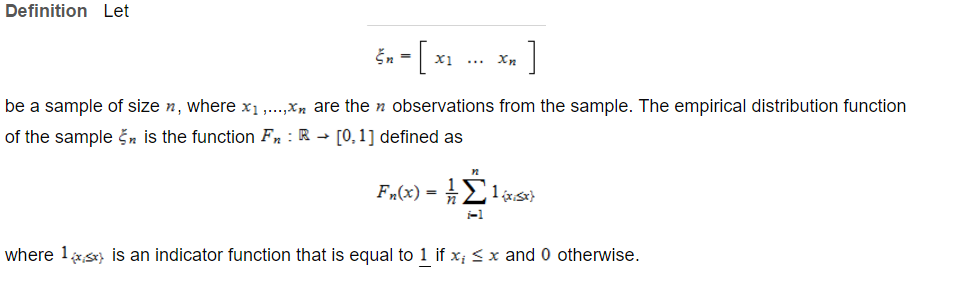
* An integral can be easily calculated via Monte Carlo simulation.
* Observing that the theoretical values and the calculated values are almost same.
* As we take more samples i.e. from 1000 to 100,000 we observe that the calculated value is approaching more closer to theoretical value.
* This means that Monte Carlo simulation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**Concept behind this problem:**

The problem uses the concept of *Empirical Distribution*. In statistics, an empirical distribution function is the distribution function associated with the empirical measure of a sample. This cumulative distribution function is a step function that jumps up by 1/n at each of the n data points. Its value at any specified value of the measured variable is the fraction of observations of the measured variable that are less than or equal to the specified value.



**Code & Code Description:**

* We first start off getting the random variable Zi^2 i.e by taking rand(1 4) and then sum of squares of the numbers for n samples.
* Then we sort the matrix(X) we received for chicdf function.
* Perform Ecdf to get empirical CDF of X and chiplot of X
* Plot both the theoretical distribution over the empirical distribution
* Calculate the
  1. The lower bound of the difference between Emp and theo values
  2. Theo and Prac values of 25th, 50th and 90th percentile

runs =[10 100 1000];

for expt = 1:3

num\_of\_samples = runs(expt);

%Generate random variable Zi^2

for iter1=1:num\_of\_samples

r\_numbers = randn(1,4);

X(iter1)=sum(r\_numbers.^2);

end

sort\_X\_series=sort(X); %For chi2cdf() function.

[emp\_prob,x]=ecdf(X);

emp\_prob=emp\_prob.';

emp\_prob = emp\_prob(2:num\_of\_samples+1);

cdfplot(X);% Empirical Distribution Function

hold on

theo\_prob=chi2cdf(sort\_X\_series,4); %Theoretical distribution using chi2cdf function

plot(1:num\_of\_samples,theo\_prob,'linewidth',2); % Overlaying the theoretical distribution over emprical distribution

hold off

% Calculating the lower bound by iterating over all the sample sizes

for iter2=1:num\_of\_samples

Lower\_Bound=max(abs(emp\_prob(iter2)- theo\_prob(iter2)));

end

Theo\_90th= chi2inv(0.9,4);

Theo\_50th=chi2inv(0.5,4);

Theo\_25th=chi2inv(0.25,4);

Prac\_25th=prctile(sort\_X\_series,25);

Prac\_50th=prctile(sort\_X\_series,50);

Prac\_90th=prctile(sort\_X\_series,90);

disp(['Summary for ',num2str(num\_of\_samples),' samples']);

disp(['The lower bound of the difference between Emp and theo values=',num2str(Lower\_Bound)]);

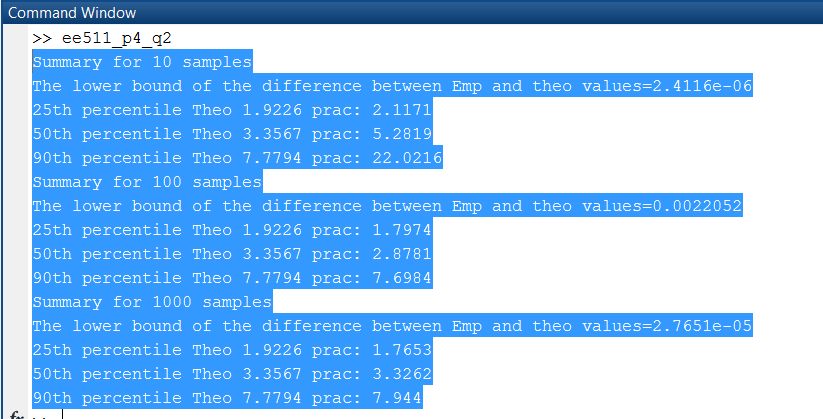
disp(['25th percentile Theo ',num2str(Theo\_25th),' prac: ',num2str(Prac\_25th)]);

disp(['50th percentile Theo ',num2str(Theo\_50th),' prac: ',num2str(Prac\_50th)]);

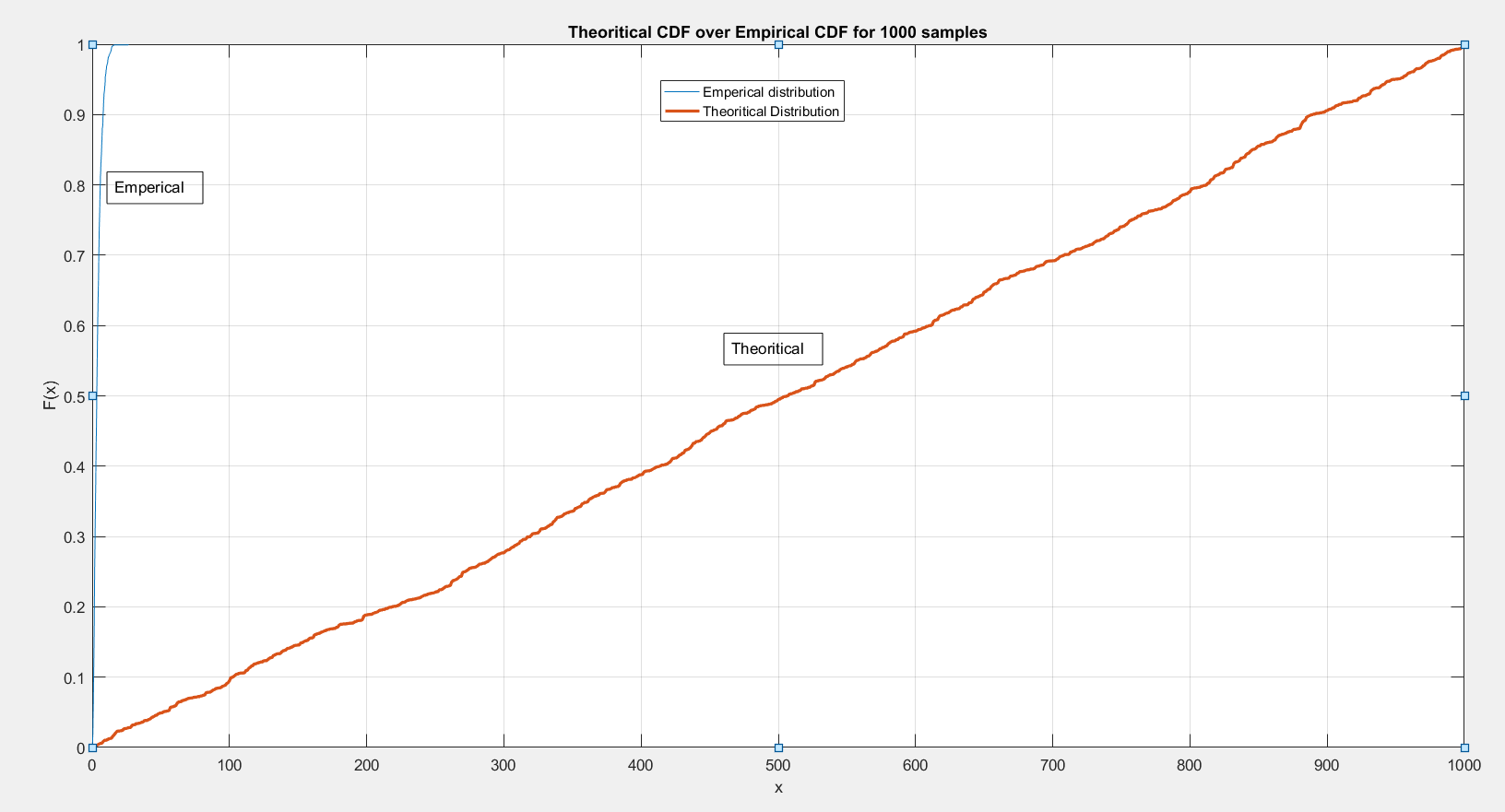
disp(['90th percentile Theo ',num2str(Theo\_90th),' prac: ',num2str(Prac\_90th)]);

end

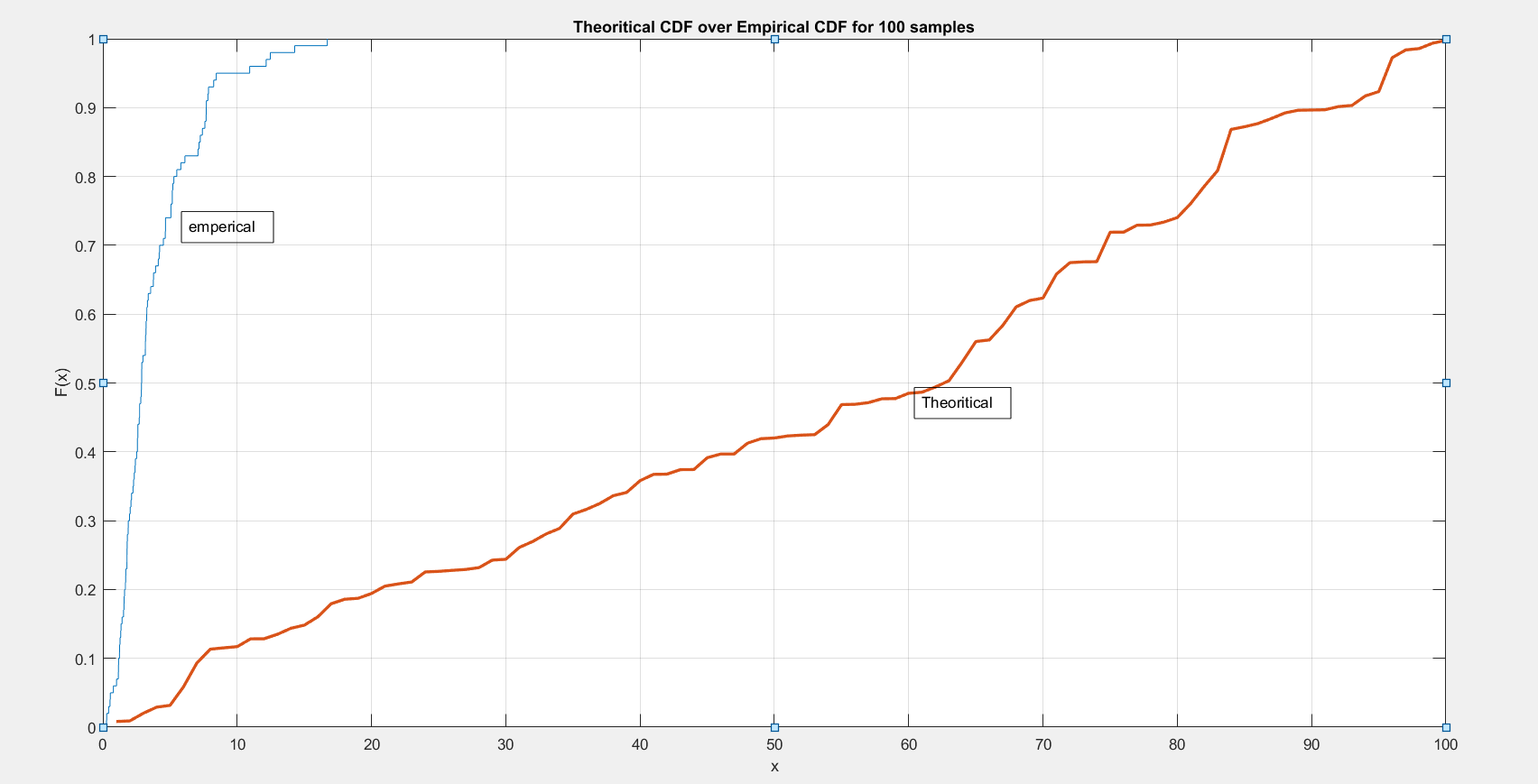
**Results:**



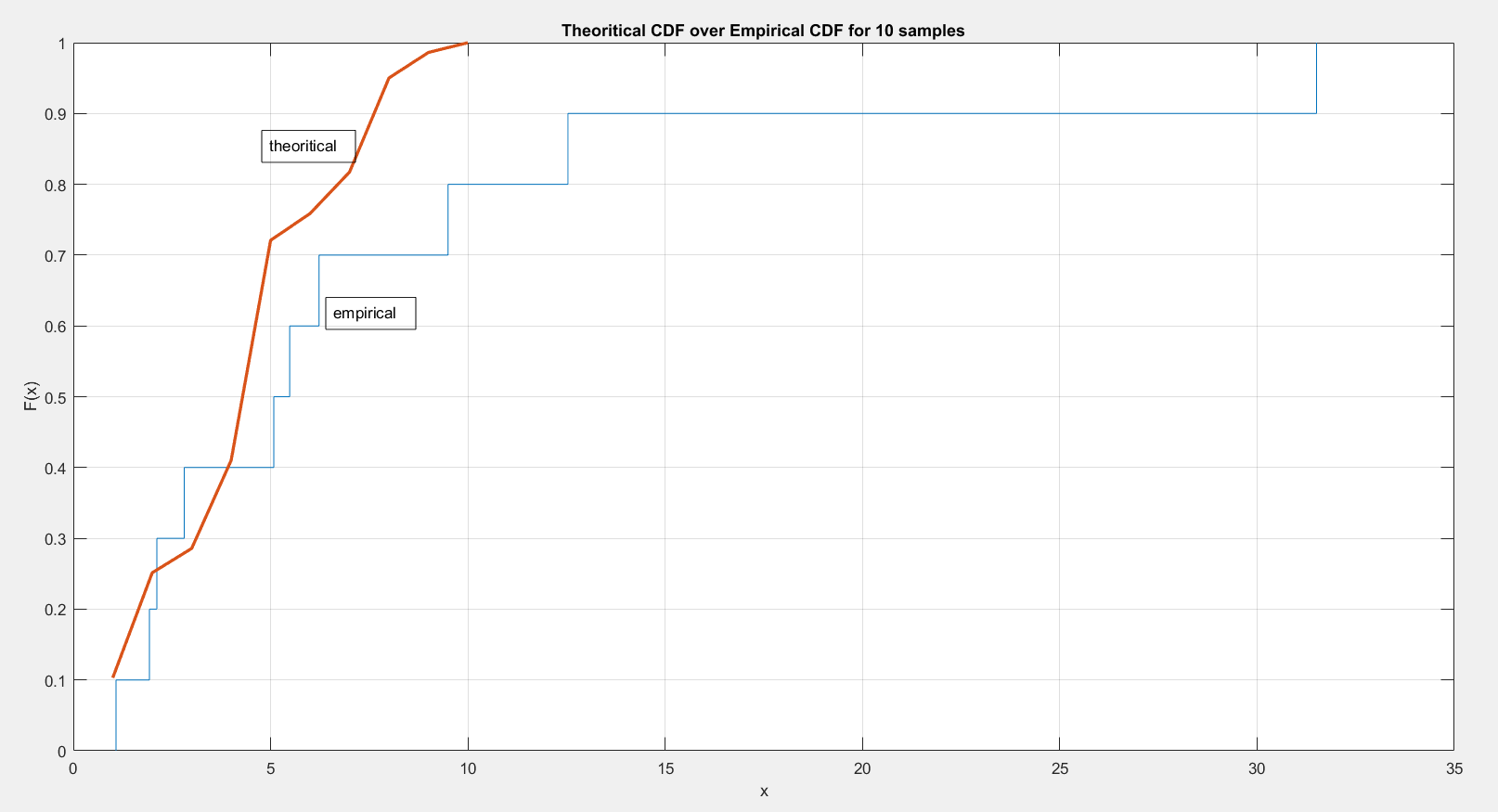
1000 Samples



100 samples



10 samples



**Observations:**

* The theoretical and the Empirical values are almost the same.
* The theoretical distribution coincides with the empirical distribution
* As we increase the number of samples, we see that the 25th,50th and 90th percentile calculated values are almost same with the theoretical values.

3) *A geyser is a hot spring characterized by an intermittent discharge of water and steam. Old Faithful is a famous cone geyser in Yellowstone National Park, Wyoming. It has a predictable geothermal discharge and since 2000 it has erupted every 44 to 125 minutes. Refer to the addendum data file that contains waiting times and the durations for 272 eruptions. Compute a 95% statistical confidence interval for the waiting time using data from only the first 15 eruptions. Compare this to a 95% bootstrap confidence interval using the same 15 data samples. Repeat these calculation using all the data samples. Comment on the relative width of the confidence intervals when using only 15 samples vs using all samples.*

**Concept behind this problem:**

|  |
| --- |
| The problem requires the concept of Importance sampling via T Distribution. The T distribution (aka, Student’s t-distribution) is a probability distribution that is used to estimate population parameters when the sample size is small and/or when the population variance is unknown. Sometimes the sample sizes are small, and often we do not know the standard deviation of the population. When either of these problems occur, statisticians rely on the distribution of the t statistic (also known as the t score), whose values are given by: |

where x is the sample mean, μ is the population mean, s is the standard deviation of the sample, and n is the sample size. The distribution of the t statistic is called the t distribution or the Student t distribution. The t distribution allows us to conduct statistical analyses on certain data sets that are not appropriate for analysis, using the normal distribution.

**Code & Code Description:**

* Here we first load the faithful data onto the matlab project by use of fullfile.
* Once we have the data of eruptions and the waiting time with us, we create two variables where in one case we run the entire code for first 15 samples and in the other case we use the entire 272 samples.
* We calculate the mean of those samples. If it is first 15 samples, its sample mean. If it is the entire 272 samples, it’s called population samples.
* We then take random integer numbers in the interval of 44-125 for 15(in loop-1) and 272 (in loop-2).
* For each sample compute the sample mean, named xbar, and the sample standard deviation named s.
* Next, confidence intervals were computed as xbar-t(s/Sqrt[5]) to xbar+t(s/Sqrt[5]).
* Get the margin\_of\_error by multiplying t\_dist\*std\_error .
* Our confidence interval is the mean of 15/272 samples +- mean(margin\_of\_error).
* This gives us the test of whether that confidence interval is good (includes the population mean) or bad (doesn't include the population mean).

function [ ] = stat\_confi\_interval()

%Loading faithful.txt file into the matlab

fullname = fullfile('D:\Usc\Sem1\511\Project4','faithful.txt');

%Initialize

loaded\_data= load(fullname);

waiting\_time\_272 = loaded\_data(:,3);

waiting\_time\_15 = waiting\_time\_272(1:15);

mean\_waiting\_time\_15 = mean(waiting\_time\_15);

mean\_waiting\_time\_272 = mean(waiting\_time\_272);

for expt=1:2

if expt ==1

waiting\_time = waiting\_time\_15;

mean\_waiting\_time = mean\_waiting\_time\_15;

len = 15;

else

waiting\_time = waiting\_time\_272;

mean\_waiting\_time = mean\_waiting\_time\_272;

len = 272;

df = len-1;

statistical\_value = tinv(0.95,df);

end

disp(['Summary for ',num2str(len),' samples:']);

for iter=1:100

r\_numbers = randi([44 125],1,len);

x\_bar(iter) = mean(r\_numbers);

samp\_std(iter) = std(r\_numbers);

std\_error(iter) = samp\_std(iter)/sqrt(len);

t\_dist(iter) = (x\_bar(iter) - mean\_waiting\_time)/(samp\_std(iter)/sqrt(len));

margin\_of\_error(iter) = t\_dist(iter)\*std\_error(iter); end;

boot\_ci = bootci(len,@mean,waiting\_time);

ci\_low = mean\_waiting\_time - mean(margin\_of\_error);

ci\_high = mean\_waiting\_time + mean(margin\_of\_error);

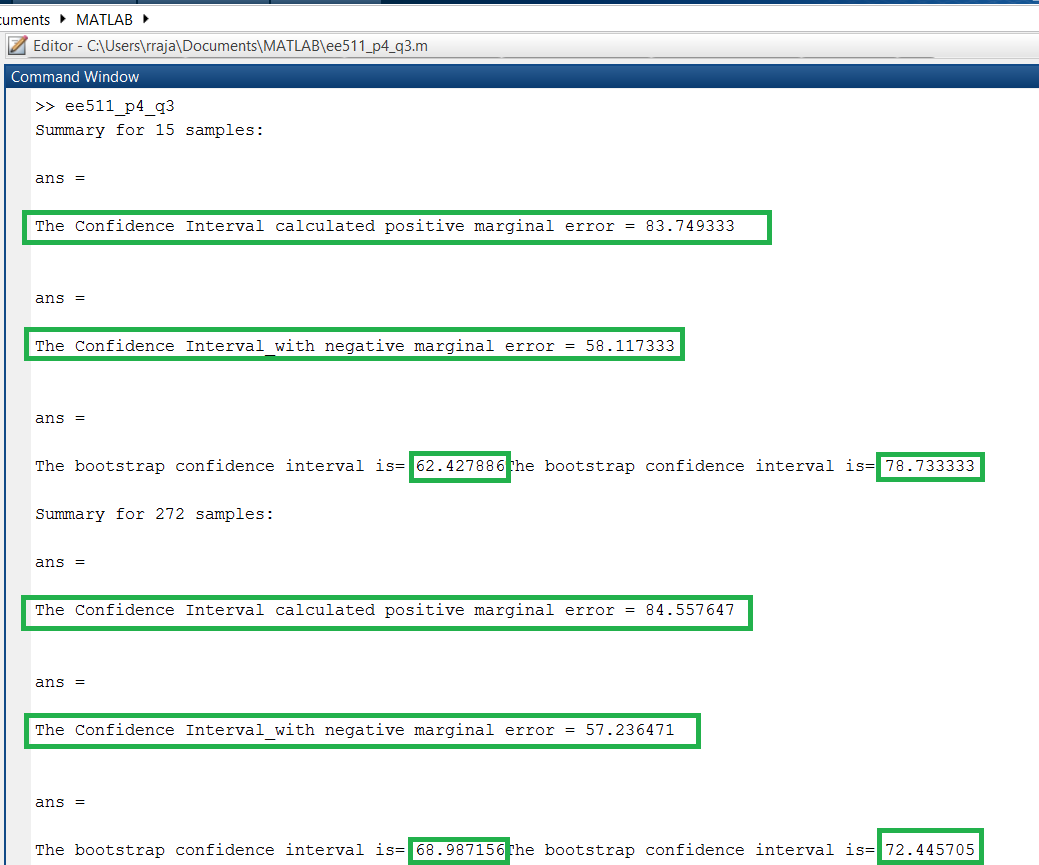
sprintf('The Confidence Interval calculated positive marginal error = %f', ci\_high)

sprintf('The Confidence Interval\_with negative marginal error = %f', ci\_low)

sprintf('The bootstrap confidence interval is= %f', boot\_ci)

end;

**Results and observations:**



* The Statistical Confidence interval obtained with 15 samples via t distribution calculation is in line with 95% bootstrap confidence interval using the same 15 data

Samples.

* The Statistical Confidence interval obtained with all 272 samples via t distribution calculation is in line with 95% bootstrap confidence interval using the same 15 data

Samples.

* We observe that the 95% statistical confidence interval for the waiting time using data

from only the first 15 eruptions is almost same as the calculation using all 272 of the data samples.

* This means that in order to get 95% confidence interval in the entire population, it is ok to perform sampling for 15 samples and not take the entire 272 samples since the values are almost similar to give complete information about the entire distribution.