Project #4 - Integrals and Intervals

Li Yicheng*

USCID:7827077047

email: l.y.c.liyicheng@gmail.com

USC Viterbi of Engineering

I. Computing Integrals and monte carlo simulation

I. Requirement

Compute the following integrals

a: $\int_{-2}^{2} e^{x+x^2} dx$ **b:** $\int_{-\infty}^{\infty} e^{-x^2} dx$ **c:** $\int_0^1 \int_0^1 e^{-(x+y)^2} dy dx$

II. Experiment

a: Monte Carlo Simulation

The basic idea is $\int_0^1 g(x)dx = E(g(u))$ where **u** is samples draw from the standard uniform distribution. Actually it is easy to qualitativly prove this. For instance we have n samples u_i draw from [0,1], then we divide the [0,1] into n pieces and assign width $1/n$ to each sample u_i, each sample has a height g(u_i), and it has a corresponding area $\frac{1}{n}g(u_i)$. And we add all the areas together to simulate the value of the integral, it would be $\sum_1^n \frac{1}{n}g(u_i)$ and this is $E(g(u))$. The simulation can be accurate if lots of samples are drawn from the standard uniform distribution.To put this simulation method into application, the following three situations are discussed.

***1** if we would like to integral $f(x)$, x range from a to b. We should first translate the range [a,b] to [0,1]. This is easy to be implemented by let $y = \frac{x-a}{b-a}$. In this case. Then compute the new dx, at last we can derive the final formular: $\int_a^b g(x)dx = (b-a)\int_0^1 g((b-a)y+a)dy$

***2** if we would like to integral $f(x)$, x has an infinite range, say x from a(a>0) to ∞, we use the same idea in *1, we translate the range $x \in [a, \infty]$ to $y \in [0, 1]$, this can be done by letting $y = \frac{1}{1+x-a}$, continue doing calculation we can finally get $\int_a^\infty g(x)dx = \int_0^1 g(\frac{1}{y} + a - 1)\frac{1}{y^2}dy$

***3** when it comes to the multiple integrals in the form like

$\int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_{n-1}}^{b_{n-1}} \int_{a_n}^{b_n} g(x_1, x_2, x_3 \cdots x_n) dx_1 dx_2 \cdots dx_n$.

It is also easy to derive the simulation by translate each range to the [0,1], $x_n \to u_n, u_n \sim u[0,1]$ compute the $x_n = T(u_n)$, Therefore: $\int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_{n-1}}^{b_{n-1}} \int_{a_n}^{b_n} g(x_1, x_2, x_3 \cdots x_n) dx_1 dx_2 \cdots dx_n$

$= \int_0^1 \int_0^1 \cdots \int_0^1 g(T_1(u_1), T_2(u_2) \cdots T_n(u_n)) dT_1(u_1) dT_2(u_2) \cdots dT_n(u_n)$

*github link: https://github.com/IAMLYCHEE/EE501-PROJ4

b.Implementation

***1** Compute $\int_{-2}^{2} e^{x+x^2} dx$

filename: integral_fi.m

```
function result = integral_fi(a,b,eval_budget)
%b is the upper bound of the integral
%a is the lower bound of the integral
%eval_budget determines the amount of samples generated to evaluate
unif_samples = rand(eval_budget,1);
x = (b - a) * unif_samples + a; %y -> x
g_x = exp(x.^2 + x); %compute g(x)
result = (b - a).*mean(g_x);
```

***2** Compute $\int_{-\infty}^{\infty} e^{-x^2} dx$

filename: integral_inf.m

```
function result = integral_inf(a,eval_budget)
%result = integral_inf(a,eval_budget)
%a : the input lower bound , a >= 0
%eval_budget: the amount of samples to get the result

y = rand(eval_budget,1)+1e-10; %y>0
x = y.^(-1) - (1 - a); %translate the range
%since this is an even symmetry function we only need to calculate half of the integral
result = 2*mean( exp(- x.^2) ./ (y.^2) );
```

***3** Compute $\int_0^1 \int_0^1 e^{-(x+y)^2} dy dx$

filename: multi_integ.m

```
function result = multi_integ(eval_budget)
%result = multi_integ(va_amount, eval_budget)
%this is multiintegral with bound 0 to 1
%eval_budget: the amount of samples to draw
u = rand(eval_budget,2);
result = mean( exp( -1 * (u(:,1) + u(:,2)).^2));
```

III. Result & Analysis

***1**

query: result = integral_fi(-2,2,10000)

result = 92.9909

theoretical result:

$$result = -\frac{e^{-\frac{1}{4}}\sqrt{\pi}i\left(\operatorname{erf}\left(\frac{5i}{2}\right)+\operatorname{erf}\left(\frac{3i}{2}\right)\right)}{2} = 93.16275329244197$$

***2**

query: result = integral_inf(0,15000)

result = 1.7801

theoretical result:

$result = \sqrt{\pi} = 1.772453850905516$

***3**

query: result = multi_integ(15000)

result = 0.4111

theoretical result:

$result = \frac{1}{2e^4}\left(-2e^3+1+2\sqrt{\pi}\left(-e^4\operatorname{erf}(1)+e^4\operatorname{erf}(2)\right)+e^4\right) \approx 0.411792894172914$

Performance Analysis

Take the first integral for example, a bunch of different sample amount were input to test the accuracy.

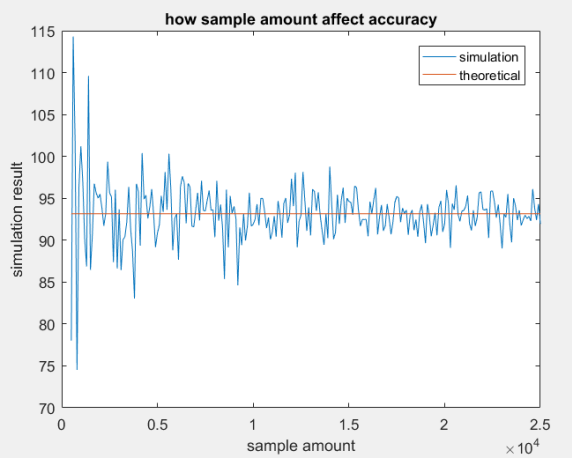


Figure 1: *accuracy test*

The sample amount vary from 500 to 25000, when the amount is small, large oscillation occurs and we may get 71.2339 as a simulation result which has a 21% difference to the theoretical value. When the sample amount get larger, it become more and more stable to derive values fit to the theoretical value.Therefore, when use such simulation method, the sample amount should not be too small.

II. EMPIRICAL DISTRIBUTION

I. Requirement

We have a random variable which is defined as $X = Z_1^2 + Z_2^2 + Z_3^2 + Z_4^2$, and Z conforms to the standard normal distribution. How to derive a distribution function of X by having 10 samples? Using the Empirical distribution to simulate the cdf of x and compare it to the theoretical cdf.

II. Experiment

Theory

The idea of empirical distribution simulation is qualitatively easy to understand. We can first define x in some range. For instance, we let $x \in [0, 15], x = nT, T = 0.01$. So we first choose $x = 0$, and find how many samples $\in (-\infty, 0)$, then choose $x = 0.01$, find how many samples $\in (-\infty, 0.01)$, continue doing so until $x = 15$. This form the empirical distribution.

Algorithm

***1** Derive several samples.

***2** Define the x range of the cdf.

***3** Iterate x in the defined range and find the number of samples in the range.

Implementation

filename: generateF10x.m

```
function Fn = generateF10x(sample,lb,ub,isplot)
%Fn = generateF10x(sample,lb,ub,isplot)
%generate the empirical distribution
%sample: the amount of samle
%lb: the lower bound of x
%ub: the upper bound of x
%isplot: whether plot or not
X = zeros(sample, 1);
for i = 1 : sample
    X(i) = sum( randn(4,1) .^2 );
end

k = length(lb: 0.05 : ub);
Fn = zeros(k , 1);
k = 1;
for x = lb : 0.05 : ub
    accsum = 0;
    for i = 1 : sample
        if X(i) < x
            accsum = accsum + 1;
        end
    end
```

```
23      Fn(k) = accsum/sample;
24      k = k + 1;
25  end
26  if isplot
27      plot((lb:0.05:ub),Fn);
28      hold on
29      x = lb : 0.05 : ub;
30      F_the = chi2cdf(x , 4);
31      plot( x , F_the)
32      hold off
33      xlabel('x')
34      ylabel('probability')
35      title('empirical distribution')
36      legend('Empirical Simulation','theoretical')
37  end
```

To further compute $||F_{10}^*(x) - F(x)||_\infty$ and the $25^{th}, 50^{th}, 90^{th}$ percentiles using empirical distribution and compare it to the theoretical value with the following script.
filename:solution2.m

```
1  %generate the theoretical chi-square distribution cdf
2  x = 0 : 0.05 : 10;
3  F_the = chi2cdf(x , 4);
4  F_the = F_the';
5  %generate simulation with 10 samples
6  Fn = generateF10x(10,0,10,true);
7  %compute differences
8  diff = abs( Fn - F_the );
9  maxDiff = max(diff);
10
11 xSize = length(x);
12 e25 = Fn(round(xSize*0.25));
13 e50 = Fn(round(xSize*0.50));
14 e90 = Fn(round(xSize*0.90));
15 the25 = F_the(round(xSize*0.25));
16 the50 = F_the(round(xSize*0.50));
17 the90 = F_the(round(xSize*0.90));
```

III. Result & Analysis

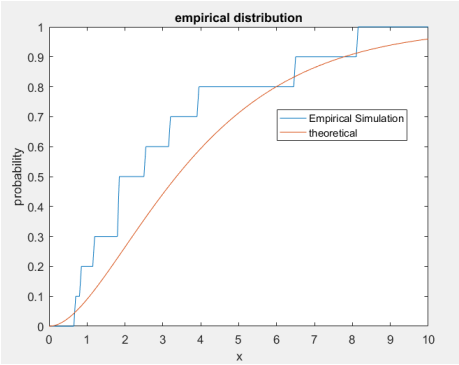


Figure 2: *Empirical Distribution*

Result:

$n^{th} percentile$	empirical	theoretical
25	0.2000	0.3464
50	0.7000	0.7127
90	0.9000	0.9389

maximum difference = 0.2422.

The max difference between empirical distribution and theoretical distribution is 0.2422. This seems to be a large difference when it comes to probability. However, when $Pr(X <= x)$ is near 1, the empirical distribution approaches theoretical distribution. That is, for the x^th percentiles, the more x approaches 1, the difference between $F_{10}^{*}(x)$ *and* $F(x)$.

If the sample amount is 100, we get another result:

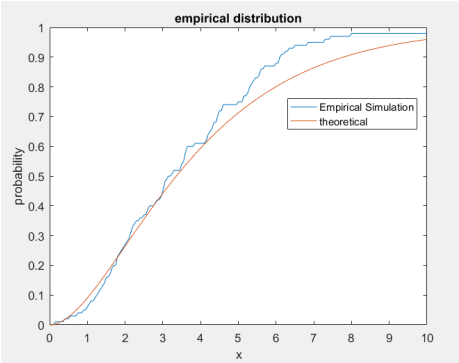


Figure 3: *Empirical Distribution*

Result:

$n^{th} percentile$	empirical	theoretical
25	0.3500	0.3464
50	0.7600	0.7127
90	0.9500	0.9389

maximum difference = 0.0663.

III. CONFIDENCE INTERVAL

I. Requirement

We have a data file recording a geyser's eruption time and waiting time. The data has the population size 272. However, we estimate the waiting time and erruption time using only the first 15 samples. Derive the confidence interval using the 15 data samples, and compare it to the confidence interval of the whole population.

II. Experiment

a:Confidence Interval

The statistical confidence interval uses the Z-distribution and conforms $(m - \frac{s}{\sqrt{n}} z_{\alpha/2}, m + \frac{s}{\sqrt{n}} z_{\alpha/2})$ for a $100(1-\alpha)\%$ confidence interval. Here m is the observed value of sample mean. and s be the standard deviation.

Generate two functions, one is to compute the confidence interval, the other to compute the bootstrap confidence interval.

b.Implementation

filename: computeCI.m

```
function [mu,var] = computeCI(data,sampleAmount)
%compute the comfidence interval for the mean and standard deviation
%data: the input data
%sampleAmount: the amount of sample from the data
dataSample = data(1:sampleAmount);
pd = fitdist(dataSample,'Normal');
CI = paramci(pd);
mu = CI(:,1);
var = CI(:,2);
```

filename: computeBootstrapCI.m

```
function [mu,var]=computeBootstrapCI(data,sampleAmount)
%compute the bootstrap confidence interval for the mean and standard deviation
%data: the input data
%sampleAmount: the amount of sample from the data
data = data(1:sampleAmount);
dataMean = mean(data);
dataVars = abs(data - dataMean);
data=sort(data);
dataVars = sort(dataVars);
mu = [data(ceil(sampleAmount * 0.025)),data(ceil(sampleAmount*0.975))];
var = [dataVars(ceil(sampleAmount * 0.025)),dataVars(ceil(sampleAmount*0.975))];
```

query script:

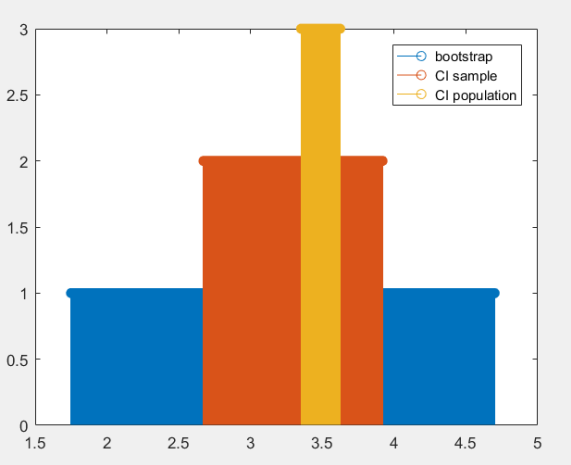
```
data = load('duration.mat');
duration = data.duration;
data = load('waiting.mat');
waiting = data.waiting;
[durationMu_15,durationVar_15] = computeCI(duration,15);
[durationMu_po,durationVar_po] = computeCI(duration,length(duration));
[waitingMu_15,waitingVar_15] = computeCI(waiting,15);
[waitingMu_po,waitingVar_po] = computeCI(waiting,length(waiting));
```

```
[durationMu_15_bootstrap,durationVar_15_bootstrap] = computeBootstrapCI(duration,15);
[durationMu_po_bootstrap,durationVar_po_bootstrap] = computeBootstrapCI(duration,length(
    duration));
[waitingMu_15_bootstrap,waitingVar_15_bootstrap] = computeBootstrapCI(waiting,15);
[waitingMu_po_bootstrap,waitingVar_po_bootstrap] = computeBootstrapCI(waiting,length(
    waiting));
```

III. Result & Analysis

Name	Value
data	*1x1 struct*
duration	*272x1 double*
durationMu_15	[2.6719;3.9190]
durationMu_15_bootstrap	[1.7500,4.7000]
durationMu_po	[3.3515;3.6240]
durationMu_po_bootstrap	[1.7500,4.9330]
durationVar_15	[0.8244;1.7758]
durationVar_15_bootstrap	[0.0375,1.5455]
durationVar_po	[1.0528;1.2463]
durationVar_po_bootstrap	[0.1122,1.7378]
waiting	*272x1 double*
waitingMu_15	[62.5571;79.3096]
waitingMu_15_bootstrap	[47,88]
waitingMu_po	[69.2742;72.5199]
waitingMu_po_bootstrap	[46,90]
waitingVar_15	[11.0738;23.8544]
waitingVar_15_bootstrap	[3.0667,23.9333]
waitingVar_po	[12.5404;14.8446]
waitingVar_po_bootstrap	[0.8971,24.8971]

The left shows the confidence interval of the mean and the deviation.If we qualitatively plot these confidence intervals, we would have the following figure:



From the figure, we can see that there exist a shift betwwen the 15 samples and the whole population, this means given the first 15 samples the estimation can be not accurate enough. The statistic confidence interval tries to gather the two sides' information to the mid using the normal distribution assumption and thus performs better than the bootstrap confidence interval result.