Rajat Kumar Mandal-226121014

Mathematica file with explanations

In[2]:=
```
(* Explain the meaning of each line *)
```

In[3]:=
```
(*xn is drawn from a uniform distribution in [0,1]*)
x[n_] := RandomReal[]
```

In[4]:=
```
(*x1*)
x[1]
```

Out[4]= 0.289361

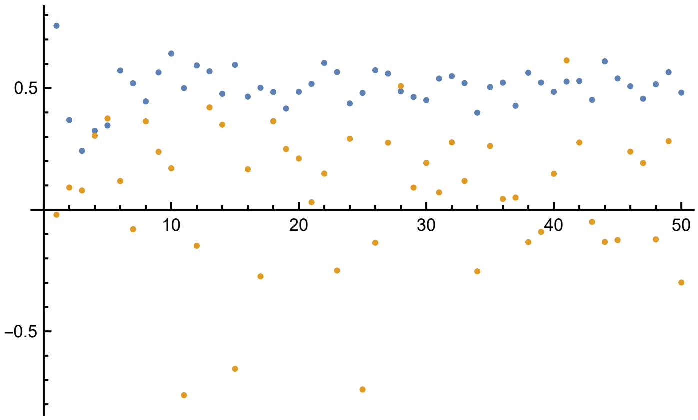
In[5]:=
```
(*x2*)
x[2]
```

Out[5]= 0.839529

In[6]:=
```
(*Average of x1, x2, x3... upto xn, in the limit n→∞,
the distribution of this average approaches to a gaussian with 0 standard deviation,
which is a dirac delta distribution*)
AvgX[n_] := Sum[x[j], {j, 1, n}] / n
(*Average of x1, x2, x3... upto xn, subtracted from mean of the uniform distribution,
scaled by √n. This is the random variate appearing in the Central Limit Theorem,
its distribution approaches towards a Gaussian with finite standard deviation*)
CLTX[n_] := (Sum[x[j], {j, 1, n}] / n - 0.5) √n
```

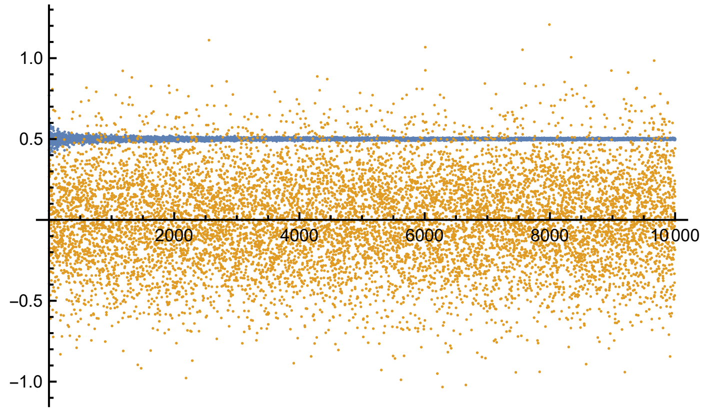
In[8]:=
```
(*As we can indeed see, the distribution of the average is being squished for
 large n but the random variate appearing in CLT has an almost constant σ*)
ListPlot[ {Table[AvgX[n], {n, 1, 50}], Table[CLTX[n], {n, 1, 50}]}]
```

Out[8]=



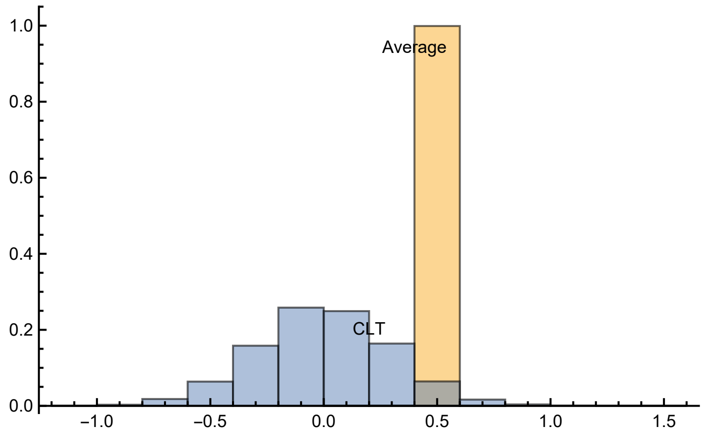
In[11]:=
```
(*As we can indeed see, the distribution of the average is being
 squished for large n (approaching a δ distribution) but the random
 variate appearing in CLT is a Gaussian with almost constant σ*)
ListPlot[ {Table[AvgX[n], {n, 1, 10^4}], Table[CLTX[n], {n, 1, 10^4}]}]
```

Out[11]=



In[12]:=
```
(*A better representation is the histogram, where we can see the distributions*)
Histogram[{Table[AvgX[n], {n, 1, 10^4}], Table[CLTX[n], {n, 1, 10^4}]},
 Automatic, "Probability", ChartLabels → {"Average", "CLT"}]
```

Out[12]=

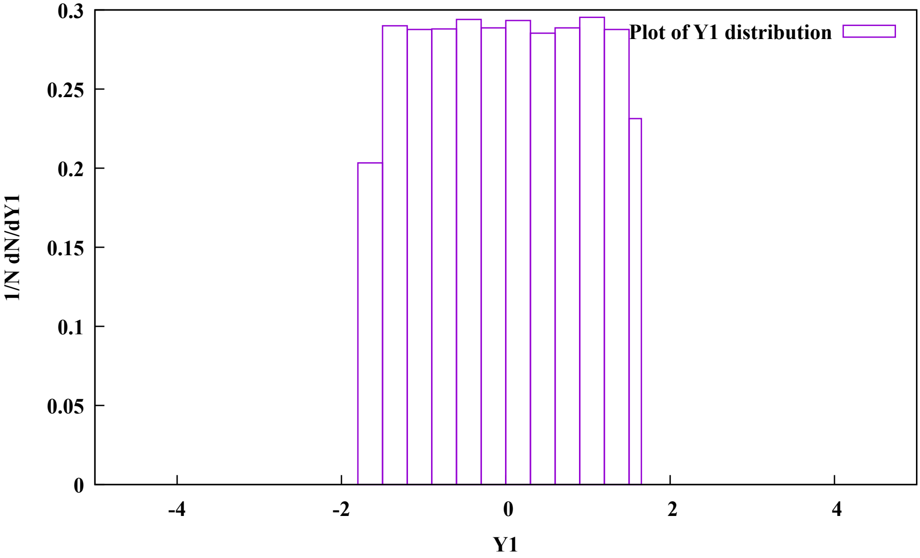


C++ code for problem 1


```cpp
#include <random>
#include <array>
#include <cmath>
#include <fstream>

//The sample size for plotting final distribution - this many numbers will be drawn
constexpr size_t samplesize = 10000;

int main()
{
    std::array<double, samplesize> Z{}; //array to store the values, in case we need

    std::random_device dev; //Responsible for getting a random seed from OS
    std::mt19937_64 rng(dev());    //Mersenne Twister engine with the seed for generating pseudo-random numbers
    std::uniform_real_distribution<double> dist(0,1); // distribution in range [0, 1]

    double sigmainverse = sqrt(12.0);   // 1/(standard deviation) for the uniform distribution
    double mean = 0.5;  //mean of the uniform distribution

    std::ofstream outfile;  //file handle to save the results in a file
    outfile.open("./output/problem1.txt", std::ios::out | std::ios::trunc);

    for(auto& Zi : Z){  //Loop through the array to store the values
        Zi = sigmainverse * (dist(rng) - mean);    //   calculate Y1 and store in the array
        outfile << Zi << std::endl; //write to the output file
    }
    outfile.close();    //when done, close the file.
}
```

Gnuplot script for problem 1


```
set xlabel 'Y1'
set ylabel '1/N dN/dY1'
set xrange [-5:5]

set ytics nomirror
set xtics nomirror

set terminal pdf font "Times New Roman-Bold"

set output 'problem1.pdf'

samplesize = 10000
binwidth = 0.3
bin(x, width) = width * floor(x / width) + width/2.0

plot 'problem1.txt' using (bin($1, binwidth)):(1.0 / (binwidth * samplesize)) smooth freq with boxes title 'Plot of Y1 distribution'

```

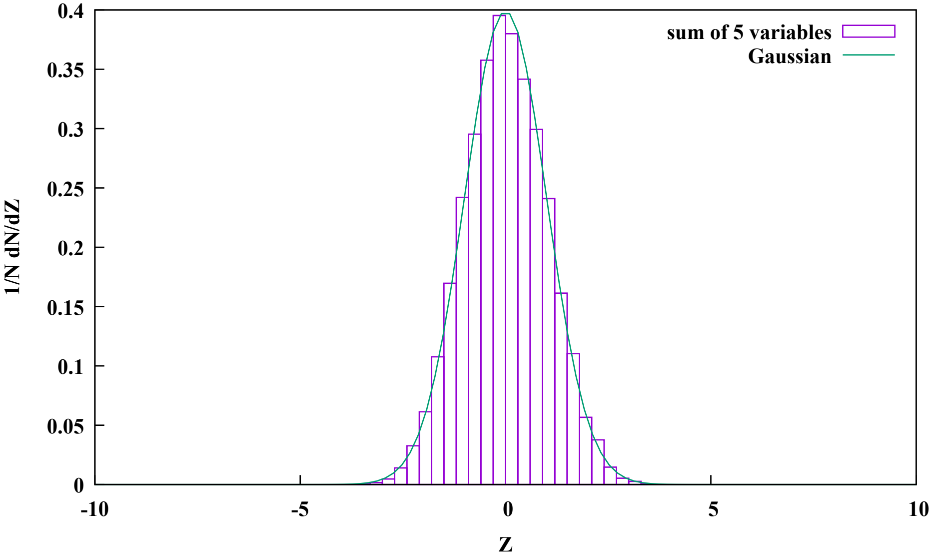

Plot of Y1 distribution
1/N dN/dY1
Y1
0.3
0.25
0.2
0.15
0.1
0.05
0
-4
-2
0
2
4

C++ code for problem 2

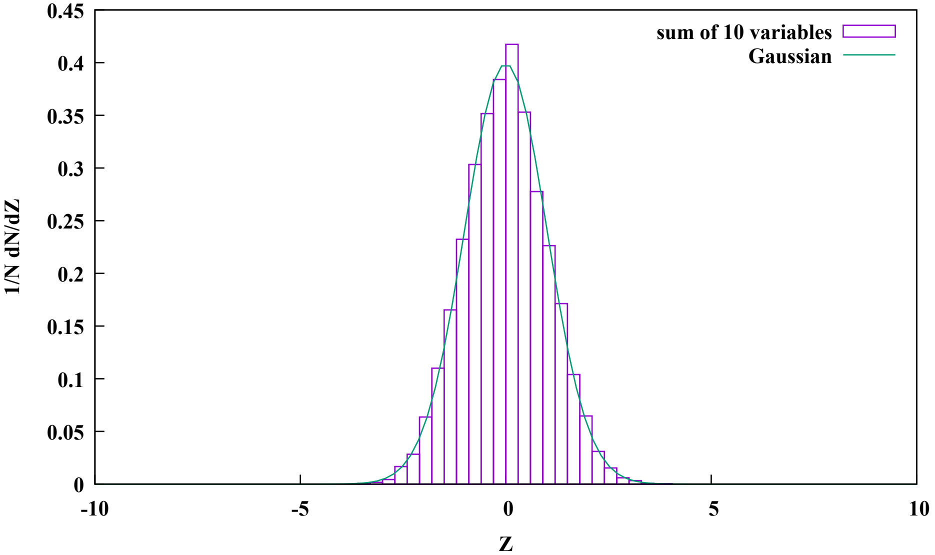

```cpp
#include <random>
#include <array>
#include <cmath>
#include <fstream>

//Constant expressions appearing in the code
constexpr size_t samplesize = 10000;    //The sample size for plotting final distribution - this many numbers will be drawn
constexpr std::array<double, 4> numvars = {5, 10, 100, 2000};   //array of the number of random variables

int main()
{
    std::array<double, samplesize> Z{}; //array to store the values, in case we need

    std::random_device dev; //Responsible for getting a random seed from OS
    std::mt19937_64 rng(dev());    //Mersenne Twister engine with the seed for generating pseudo-random numbers
    std::uniform_real_distribution<double> dist(0,1); // distribution in range [0, 1]

    double sigmainverse = sqrt(12.0);   // 1/(standard deviation) for the uniform distribution
    double mean = 0.5;  //mean of the uniform distribution

    std::ofstream outfile;  //file handle to save the results in a file
    outfile.open("./output/problem2_2000.txt", std::ios::out | std::ios::trunc);

    for(auto& Zi : Z){  //Loop through the array to store the values
        for (size_t i = 0; i < numvars[3]; i++) {   //loop through the number of variables to sum over
            Zi += sigmainverse * (dist(rng) - mean);    //calculate Yi and add to Z
        }
        Zi /= sqrt(numvars[3]); //divide by sqrt(n)
        outfile << Zi << std::endl; //write in the output file.
    }
    outfile.close();    //when done, close the file.
}
```

Gnuplot script for problem 2

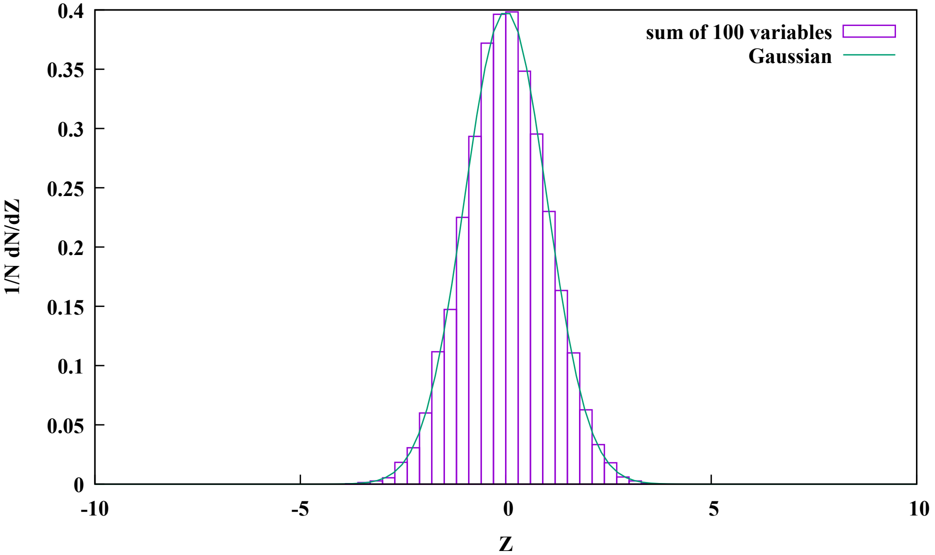

```
set xlabel 'Z'
set ylabel '1/N dN/dZ'
set xrange [-10:10]

set ytics nomirror
set xtics nomirror

set terminal pdf font "Times New Roman-Bold"

set output 'problem2_2000.pdf'

samplesize = 10000
binwidth = 0.3
bin(x, width) = width * floor(x / width) + width/2.0

plot 'problem2_2000.txt' using (bin($1, binwidth)):(1.0 / (binwidth * samplesize)) smooth freq with boxes title 'sum of 2000 variables', exp(-x * x/2)/sqrt(2 * pi) title "Gaussian"

```

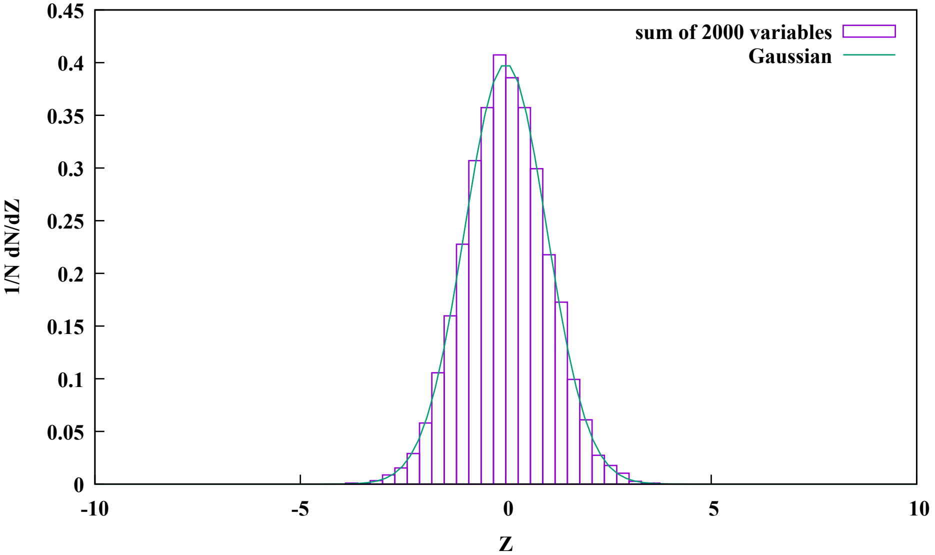


sum of 5 variables
Gaussian
1/N dN/dZ
Z
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10
-5
0
5
10



sum of 10 variables
Gaussian
1/N dN/dZ
Z
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10
-5
0
5
10


sum of 100 variables
Gaussian
1/N dN/dZ
Z
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10
-5
0
5
10



sum of 2000 variables
Gaussian
1/N dN/dZ
Z
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-10
-5
0
5
10