Assignment 4

Chao Cheng

February 20, 2019

1 Question 1

It has been proposed that an approximate method for generating a standard Gaussian X is using a normalized sum of n uniformly distributed random variables. In other words,

$$X = \frac{S_n - a_n}{b_n}. \tag{1}$$

where $S_n = U_1 + U_2 + \cdots + U_n$. Fund a_n and b_n so that X has the correct first two moments. Implement this method for $n = 12$ and check the accuracy by plotting a histogram of simulated values and comparing with the theoretical density. Compare the efficiency of this method by a direct timing comparison with the Box-Muller method.

1.1 Answer

The first two moments are mean and variance, hence in order to have right moments of a standard Gaussian distribution.

$$E(X) = E(\frac{S_n - a_n}{b_n}) = \frac{E(S_n) - a_n}{b_n} = 0 \tag{2}$$

Hence,

$$a_n = E(S_n) = \frac{n}{2} \tag{3}$$

Similarly,

$$VarX = Var(\frac{S_n - a_n}{b_n}) = \frac{1}{b_n^2}Var(S_n - a_n) = \frac{1}{b_n^2}Var(S_n) = 1 \tag{4}$$

Therefore,

$$b_n = \sqrt{Var(S_n)} = \sqrt{\frac{n}{12}} \tag{5}$$

1.2 Code

```
#include<iostream>
#include<cmath>
#include<chrono>
#include<fstream>
#define pis 8*atan(1)//2
#define an 6.0

using namespace std;
using namespace:: chrono;

//since the array is short, so we used a naive search method
int N = 1000000;//# of realization

void boxMuller(int n){
    double u1, u2;//u(0,1)
    double x, y;  //gsn(0,1)
    ofstream myfile;
    myfile.open("problem1a.csv");
    srand(1);     //set up the random seed
    for(int i= 0;i<n;i++){
        u1 = ((double)rand()/(RAND_MAX));
        u2 = ((double)rand()/(RAND_MAX));
        x= sqrt(-2*log(u1))*cos(pis*u2);
        y= sqrt(-2*log(u1))*sin(pis*u2);
        myfile<<x<<"\n";
    }
     myfile.close();
}
void appMethod(int n){
    double u,s,x; //u(0,1), sum of u, gaussian approximation
    int i;
    ofstream myfile;
    myfile.open("problem1b.csv");
    i = 0;
```

```
    srand(1);     //set up the random seed
    while(i < n){
        s = 0;    // initialize the sum
        for(int j=0;j<12;j++){
            u = ((double)rand()/(RAND_MAX));
            s = s + u;
        }
        x = s - an;
        myfile<<x<<"\n";
        i=i+1;
    }
    myfile.close();
}
int main(){
    cout<<"N="<<N<<endl;
    high_resolution_clock::time_point t1 = high_resolution_clock::now();
    boxMuller(N);
    high_resolution_clock::time_point t2 = high_resolution_clock::now();
    auto duration1 = duration_cast<microseconds>( t2 - t1 ).count();
    cout<<"Box-Muller␣method␣needs␣"<<duration1<<"␣microseconds"<<endl;
    high_resolution_clock::time_point t3 = high_resolution_clock::now();
    appMethod(N);
    high_resolution_clock::time_point t4 = high_resolution_clock::now();
    auto duration2 = duration_cast<microseconds>( t4 - t3 ).count();
    cout<<"Approximate␣method␣needs␣"<<duration2<<"␣microseconds"<<endl;
    if(duration1<duration2){
        cout<<"Box-Muller␣method␣is␣more␣efficient!"<<endl;
    }else{
        cout<<"Approximate␣method␣is␣more␣efficient!"<<endl;
    }
    return 0;
}
```

1.3 Results

```
N=1000000
Box-Muller method needs 475149 microseconds
Approximate method needs 366185 microseconds
Approximate method is more efficient!
```

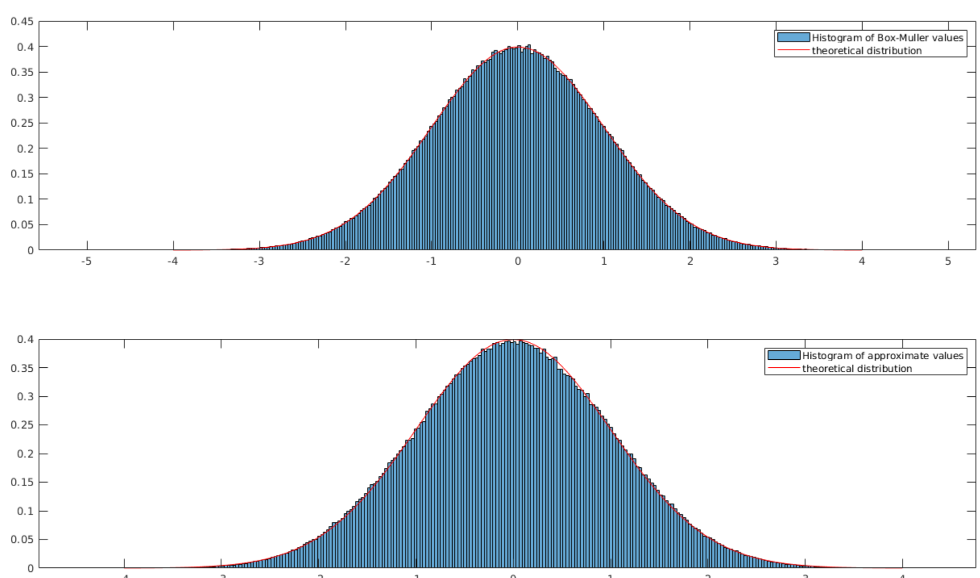

Figure 1: Comparison between normalized histogram and the pdf

2 Question 2

Give an algorithm for simulating a multivariate Gaussian with mean vector 0 and covariance matrix

$$\begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 3 & 2 \end{bmatrix}$$

Implement your algorithm and check its validity by calculating appropriate statistics.

2.1 Answer

Since the

$$\Sigma = AA^T = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 3 & 2 \end{bmatrix}$$

Hence,

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 0.5 & 1.25 & 0.433 \end{bmatrix}$$

2.2 Code

```
#include<iostream>
#include<cmath>
#define pis 8*atan(1)//2
using namespace std;
double boxMuller(){
    double u1, u2;//u(0,1)
    double x, y;  //gsn(0,1)
    u1 = ((double)rand()/(RAND_MAX));
    u2 = ((double)rand()/(RAND_MAX));
    x= sqrt(-2*log(u1))*cos(pis*u2);
    y= sqrt(-2*log(u1))*sin(pis*u2);
    return x;
}
int main(){
    double z[3];            // std gsn vector
    double x[3];            // multivariable gsn
    double mnVect[3]={0};   // estimate mean vector
    double ceMat[3][3]={0}; // estimate covariance matrix
    int N = 100000;        // total # of realization
    double A[3][3]={{2.0,0.0,0.0},{1.0,2.0,0},{0.5,1.25,0.433}};
    srand(1);              //set up the random seed
    for(int i=0;i<N;i++){
        //generate std gsn
        for(int j=0;j<3;j++){
            z[j]=boxMuller();
            x[j]=0;
        }
        for(int j=0;j<3;j++){
            for(int k=0;k<3;k++){
                x[j] = x[j]+A[j][k]*z[k];
            }
            mnVect[j] = mnVect[j] + x[j]; //estimate mean vector
        }
        for(int j=0;j<3;j++){
            for(int k=0;k<3;k++){       //estimate cov matrix
                ceMat[j][k] = ceMat[j][k] + x[j]*x[k];
            }
        }
    }
    for(int i=0;i<3;i++){
        mnVect[i] = mnVect[i]/N;
    }
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            ceMat[i][j] = ceMat[i][j]/N - mnVect[i]*mnVect[j];
        }
    }
    cout<<"Total␣realization␣N␣=␣"<<N<<endl;
    cout<<"Estimate␣Mean␣Vector␣is␣[␣";
    for(int i=0;i<3;i++){
        cout<<mnVect[i]<<"␣";
    }
```

```
    cout<<"]."<<"\n";
    cout<<"Estimate Covariance Matrix is "<<"\n";
    for(int j=0;j<3;j++){
        for(int k=0;k<3;k++){
            cout<<ceMat[j][k]<<" ";
        }
        cout<<"\n";
    }
}
```

2.3 Results

```
Total realization N = 100000
Estimate Mean Vector is [ -0.00116597 -0.00903771 -0.00546775 ].
Estimate Covariance Matrix is
4.0003 2.00137 1.0025
2.00137 5.01471 3.00971
1.0025 3.00971 2.00571
```

3 Question 3

Use Monte Carlo integration to approximate

$$\int_0^\infty \frac{x}{(1+x^3)^2}dx \tag{6}$$

3.1 Answer

Let

$$u = \frac{1}{1+x^3},$$

hence

$$du = \frac{-3x^2}{(1+x^3)^2}dx.$$

We can rewrite the original integral to be

$$\int_0^1 \frac{1}{3x}du = \int_0^1 (\frac{u}{1-u})^{1/3}du \tag{7}$$

where u follows $u(0,1)$ distribution.

3.2 Code

```
#include<iostream>
#include<cmath>
#define oneThird 1.0/3.0
using namespace std;
int main(){
    int N=1000000;
    double u;           //u~u(0,1)
    double integral;    //integration
    srand(1);           //set up the random seed
    integral = 0.0;
    for(int i=0;i<N;i++){
        u = ((double)rand()/(RAND_MAX));
        integral = integral + pow(u/(1-u),oneThird);
    }
     integral = integral * oneThird / N;
    cout<<"The integration via Monte Carlo simulation is "<<integral<<endl;
}
```

3.3 Results

```
The integration via Monte Carlo simulation is 0.403033
```

While the exact solution is

$$\int_0^\infty \frac{x}{(1+x^3)^2}dx = -\frac{1}{9} + \frac{2\pi + \sqrt{3}}{9\sqrt{3}} \approx 0.40307 \tag{8}$$

4 Question 4

Use Monte Carlo integration to approximate

$$\int_{-\infty}^{\infty} e^{-x^4} dx \tag{9}$$

4.1 Answer

We consider an exponential distribution because of the original integration is for an even function, for Monte Carlo integration as

$$f(x) = e^{-x} \tag{10}$$

Hence, rewrite the integral to be

$$\int_{-\infty}^{\infty} e^{-x^4} dx = 2\int_{0}^{\infty} e^{-x^4} dx = 2\int_{0}^{\infty} e^{(-x^4+x)} e^{-x} dx \tag{11}$$

4.2 Code

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
    int N=1000000;
    double u,x;          //u~u(0,1),exp(1)
    double integral;  //integration
    srand(100);          //set up the random seed
    integral = 0.0;
    for(int i=0;i<N;i++){
        u = ((double)rand()/(RAND_MAX));
        x = -log(u);
        integral = integral + exp(-pow(x,4.0) + x);
    }
    integral = integral * 2.0/ N;
    cout<<"The integration via Monte Carlo simulation is "<<integral<<endl;
}
```

4.3 Results

```
The integration via Monte Carlo simulation is 1.81345
```

While the exact solution is

$$\int_{-\infty}^{\infty} e^{-x^4} dx = \frac{\Gamma(1/4)}{2} \approx 1.8128 \tag{12}$$

5 Question 5

Use Monte Carlo integration to approximate

$$\int_0^3 \int_0^2 e^{(x+y)^2} dydx \tag{13}$$

5.1 Answer

$$\int_0^3 \int_0^2 e^{(x+y)^2} dydx \approx \frac{6}{n} \sum_{i=1}^{n} e^{(x_i+y_i)^2} \tag{14}$$

where x_i, y_i follow uniform distribution $u(0,3), u(0,2)$.

5.2 Code

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
    int N = 10000000;
    double u1,u2;      //u1~u(0,2),u2~u(0,3)
    double integral;   //integration
    srand(1);          //set up the random seed
    integral = 0.0;
    for(int i=0;i<N;i++){
        u1 = ((double)rand()/(RAND_MAX))*2.0;
        u2 = ((double)rand()/(RAND_MAX))*3.0;
        integral = integral + exp((u1+u2)*(u1+u2));
    }
    integral = integral * 6.0/ N;
    cout<<"The integration via Monte Carlo simulation is "<<integral<<endl;
}
```

5.3 Results

```
The integration via Monte Carlo simulation is 7.67763e+08
```

While checking the solution with Mathematica, the integral is approximately 7.68319×10^8.