Assignment 8

Chao Cheng

April 9, 2019

1 Question 1

In Homeworks 5 and 7, you were asked to use Monte Carlo integration to approximate

$$\int_0^1 e^{x^2} dx$$

Use control variates in a simulation to approximate this integral when using $N = 10^i$ realizations for $i = 2, \cdots, 6$. Compare the variance of your estimator when using control variates with the variance of your previous results obtained without the use of a variance reduction technique.

1.1 Code

```
#include<iostream>
#include<cmath>
#include <tuple>
#define pis 8.0*atan(1)
using namespace std;
/* The control variate is u, therefore x + c(u-0.5)*/
tuple<double,double,double,double> MCIntegrelVar(int n){
    double u,us[100];           // u(0,1),
    double x, y;        //realization of exp(u^2)
    double est, estA;   //est.(integral), Control est.
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of control
    double co_xy,yvar,ybar;//covariance, y mean, variance of y
    double xbar,copt;

    u = ((double)rand()/(RAND_MAX));
    us[0] = u;
    x = exp(u*u);
    y = u;
    ybar = y;
    xbar = x;
    yvar = 0;
    co_xy = 0;
    for(int i = 1; i < 100; i++)
    {
         u = ((double)rand()/(RAND_MAX));
         us[i] = u;
         x = exp(u*u);
         y = u;
         yvar = yvar*(i - 1)/i  + (ybar - y)*(ybar - y)/(i + 1);
         ybar = ybar + (y - ybar)/(i + 1);
         xbar = xbar + (x - xbar)/(i + 1);
         co_xy = co_xy*(i - 1)/i + (x-xbar)*(y-ybar)/i; //recursive covariance
    }
    copt = -co_xy/yvar; // c_opt

    est = exp(us[0]*us[0]);
    estA = est + copt*(us[0]-0.5);
    mean = est;
    meanA = estA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < 100; i++)
    {
         x = exp(us[i]*us[i]);
         y = x + copt*(us[i]-0.5);
         est = est + (x - est)/(i+1);
         estA = estA + (y - estA)/(i+1);
         svs = svs *(i - 1)/i  + (mean - est)*(mean - est)/(i + 1);
         svsA = svsA *(i - 1)/i  + (meanA - estA)*(meanA - estA)/(i + 1);
         mean = mean + (est - mean)/(i+1);
```

```
        meanA = meanA + (estA - meanA)/(i+1);

    }

    for(int i = 100;i < n;i++){
        u = ((double)rand()/(RAND_MAX));
        x = exp(u*u);
        y = x + copt*(u-0.5);
        est = est + (x - est)/(i+1);
        estA = estA + (y-estA)/(i+1);
        svs = svs *(i - 1)/i  + (mean - est)*(mean - est)/(i + 1);
        svsA = svsA *(i - 1)/i  + (meanA - estA)*(meanA - estA)/(i + 1);
        mean = mean + (est - mean)/(i+1);
        meanA = meanA + (estA - meanA)/(i+1);
    }
    return make_tuple(mean, meanA, svs,svsA);
}
int main(){
    int N[5]={100,1000,10000,100000,1000000};
    for(int i=0;i<5;i++){
        srand(1);       //set up the same random seed for comparison
        auto Var = MCIntegrelVar(N[i]);
        cout<<"For N = "<<N[i]<<endl;
        cout<<"Mean Value of crude MC integrel is "<<get<0>(Var)<<endl;
        cout<<"Mean Value of control MC Integrel is "<<get<1>(Var)<<endl;
        cout<<"Variance of crude MC Integrel is "<<get<2>(Var)<<endl;
        cout<<"Variance of control MC Integrel is "<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

1.2 Results

```
For N = 100
Mean Value of crude MC integrel is 1.52395
Mean Value of control MC Integrel is 1.45412
Variance of crude MC Integrel is 0.00599921
Variance of control MC Integrel is 0.000353867

For N = 1000
Mean Value of crude MC integrel is 1.48865
Mean Value of control MC Integrel is 1.45789
Variance of crude MC Integrel is 0.000991057
Variance of control MC Integrel is 7.71649e-05

For N = 10000
Mean Value of crude MC integrel is 1.46457
Mean Value of control MC Integrel is 1.46151
Variance of crude MC Integrel is 0.000167223
Variance of control MC Integrel is 1.19553e-05

For N = 100000
Mean Value of crude MC integrel is 1.46105
Mean Value of control MC Integrel is 1.46267
Variance of crude MC Integrel is 2.02715e-05
Variance of control MC Integrel is 1.39934e-06

For N = 1000000
Mean Value of crude MC integrel is 1.46238
Mean Value of control MC Integrel is 1.46246
Variance of crude MC Integrel is 2.28581e-06
Variance of control MC Integrel is 1.57591e-07
```

2 Question 2

Suppose that one wished to use a simulation to estimate $\sqrt{1-U^2}$ where U is uniformly distributed on (0,1). Compare the effectiveness of using U and U^2 separately as control variates with the raw estimate in a simulation study using $N = 10^i$ realizations for $i = 2, \cdots, 6$. Also, use U and U^2 together in a multiple control and compare with your previous results.

2.1 The control variate is u

2.1.1 Code

```
#include<iostream>
#include<cmath>
#include <tuple>
using namespace std;
/* The control variate is u, therefore x + c(u-0.5)*/
tuple<double,double,double,double> Control_u(int n){
    double u,us[100];  //u(0,1),
    double x, y;       //x = sqrt(1-u^2),y = u
    double est, estA;  //est.(integral), Control est.
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of control
    double co_xy,yvar,ybar;//covariance, y mean, variance of y
    double xbar,copt;

    u = ((double)rand()/(RAND_MAX));
    us[0] = u;
    x = sqrt(1-u*u);
    y = u;
    ybar = y;
    xbar = x;
    yvar = 0;
    co_xy = 0;
    for(int i = 1; i < 100; i++)
    {
         u = ((double)rand()/(RAND_MAX));
         us[i] = u;
         x = sqrt(1-u*u);
         y = u;
         yvar = yvar*(i - 1)/i  + (ybar - y)*(ybar - y)/(i + 1);
         ybar = ybar + (y - ybar)/(i + 1);
         xbar = xbar + (x - xbar)/(i + 1);
         co_xy = co_xy*(i - 1)/i + (x-xbar)*(y-ybar)/i; //recursive covariance
    }
    copt = -co_xy/yvar; // c_opt

    est = sqrt(1-us[0]*us[0]);
    estA = est + copt*(us[0]-0.5);
    mean = est;
    meanA = estA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < 100; i++)
    {
         x = sqrt(1-us[i]*us[i]);
         y = x + copt*(us[i]-0.5); //recycle y as the unbiased estimator
         svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i + 1);
         svsA = svsA *(i - 1)/i  + (meanA - y)*(meanA - y)/(i + 1);
         mean = mean + (x - mean)/(i+1);
         meanA = meanA + (y - meanA)/(i+1);

    }

    for(int i = 100;i < n;i++){
         u = ((double)rand()/(RAND_MAX));
         x = sqrt(1-u*u);
         y = x + copt*(u-0.5);
         svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i + 1);
         svsA = svsA *(i - 1)/i  + (meanA - y)*(meanA - y)/(i + 1);
         mean = mean + (x - mean)/(i+1);
         meanA = meanA + (y - meanA)/(i+1);
    }
    return make_tuple(mean, meanA, svs,svsA);
}
int main(){
```

```
    int N[5]={100,1000,10000,100000,1000000};
    for(int i=0;i<5;i++){
        srand(1);       //set up the same random seed for comparison
        auto Var = Control_u(N[i]);
        cout<<"The control variate is u, and N = "<<N[i]<<endl;
        cout<<"Value of crude MC estimator is "<<get<0>(Var)<<endl;
        cout<<"Value of control MC estimator is "<<get<1>(Var)<<endl;
        cout<<"Variance of crude MC estimator is "<<get<2>(Var)<<endl;
        cout<<"Variance of control MC estimator is "<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

2.1.2 Results

```
The control variate is u, and N = 100
Value of crude MC estimator is 0.75306
Value of control MC estimator is 0.787076
Variance of crude MC estimator is 0.0538266
Variance of control MC estimator is 0.00730882

The control variate is u, and N = 1000
Value of crude MC estimator is 0.784426
Value of control MC estimator is 0.790329
Variance of crude MC estimator is 0.0459687
Variance of control MC estimator is 0.00662809

The control variate is u, and N = 10000
Value of crude MC estimator is 0.787352
Value of control MC estimator is 0.785269
Variance of crude MC estimator is 0.0498439
Variance of control MC estimator is 0.00765795

The control variate is u, and N = 100000
Value of crude MC estimator is 0.785401
Value of control MC estimator is 0.785305
Variance of crude MC estimator is 0.0499881
Variance of control MC estimator is 0.00756134

The control variate is u, and N = 1000000
Value of crude MC estimator is 0.785405
Value of control MC estimator is 0.78541
Variance of crude MC estimator is 0.0498311
Variance of control MC estimator is 0.00754821
```

2.2 The control variate is u^2

2.2.1 Code

```
#include<iostream>
#include<cmath>
#include <tuple>
#define onethird 1.0/3.0
using namespace std;
/* The control variate is u, therefore x + c(u-0.5)*/
tuple<double,double,double,double> Control_u2(int n){
    double u,us[100];  //u(0,1),
    double x, y,u2;       //x = sqrt(1-u^2),y, u*u
    double est, estA;  //est.(integral), Control est.
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of control
    double co_xy,yvar,ybar;//covariance, y mean, variance of y
    double xbar,copt;

    u = ((double)rand()/(RAND_MAX));
    us[0] = u;
    y = u*u;
    x = sqrt(1-y);
    ybar = y;
    xbar = x;
    yvar = 0;
    co_xy = 0;
    for(int i = 1; i < 100; i++)
    {
         u = ((double)rand()/(RAND_MAX));
         us[i] = u;
```

```
        y = u*u;
        x = sqrt(1-y);
        yvar = yvar*(i - 1)/i  + (ybar - y)*(ybar - y)/(i + 1);
        ybar = ybar + (y - ybar)/(i + 1);
        xbar = xbar + (x - xbar)/(i + 1);
        co_xy = co_xy*(i - 1)/i + (x-xbar)*(y-ybar)/i; //recursive covariance
    }
    copt = -co_xy/yvar; // c_opt

    u2 = us[0]*us[0];
    est = sqrt(1-u2);
    estA = est + copt*(u2-onethird);
    mean = est;
    meanA = estA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < 100; i++)
    {
        u2 = us[i]*us[i];
        x = sqrt(1-u2);
        y = x + copt*(u2 - onethird); //recycle y as the unbiased estimator
        svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i + 1);
        svsA = svsA *(i - 1)/i  + (meanA - y)*(meanA - y)/(i + 1);
        mean = mean + (x - mean)/(i+1);
        meanA = meanA + (y - meanA)/(i+1);

    }

    for(int i = 100;i < n;i++){
        u = ((double)rand()/(RAND_MAX));
        u2 = u*u;
        x = sqrt(1-u2);
        y = x + copt*( u2 - onethird);
        svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i + 1);
        svsA = svsA *(i - 1)/i  + (meanA - y)*(meanA - y)/(i + 1);
        mean = mean + (x - mean)/(i+1);
        meanA = meanA + (y - meanA)/(i+1);
    }
    return make_tuple(mean, meanA, svs,svsA);
}
int main(){
    int N[5]={100,1000,10000,100000,1000000};
    for(int i=0;i<5;i++){
        srand(1);       //set up the same random seed for comparison
        auto Var = Control_u2(N[i]);
        cout<<"The control variate is u^2, and N = "<<N[i]<<endl;
        cout<<"Value of crude MC estimator is "<<get<0>(Var)<<endl;
        cout<<"Value of control MC estimator is "<<get<1>(Var)<<endl;
        cout<<"Variance of crude MC estimator is "<<get<2>(Var)<<endl;
        cout<<"Variance of control MC estimator is "<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

2.2.2 Results

```
The control variate is u^2, and N = 100
Value of crude MC estimator is 0.75306
Value of control MC estimator is 0.786015
Variance of crude MC estimator is 0.0538266
Variance of control MC estimator is 0.00171418

The control variate is u^2, and N = 1000
Value of crude MC estimator is 0.784426
Value of control MC estimator is 0.788285
Variance of crude MC estimator is 0.0459687
Variance of control MC estimator is 0.00141302

The control variate is u^2, and N = 10000
Value of crude MC estimator is 0.787352
Value of control MC estimator is 0.785148
Variance of crude MC estimator is 0.0498439
Variance of control MC estimator is 0.00173746

The control variate is u^2, and N = 100000
```

```
Value of crude MC estimator is 0.785401
Value of control MC estimator is 0.785276
Variance of crude MC estimator is 0.0499881
Variance of control MC estimator is 0.00169351

The control variate is u^2, and N = 1000000
Value of crude MC estimator is 0.785405
Value of control MC estimator is 0.785387
Variance of crude MC estimator is 0.0498311
Variance of control MC estimator is 0.0016799
```

2.3 Multiple Control variates u, u^2

2.3.1 Code

```
#include<iostream>
#include<cmath>
#include <tuple>
#define onethird 1.0/3.0
using namespace std;
/* The control variate is u, therefore x + c(u-0.5)*/
tuple<double,double,double,double> Control_uu2(int n){
    double u,us[100];  //u(0,1),
    double x,y1,y2,u2,y; //x = sqrt(1-u^2),y1=u,y2=u*u
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of control
    double co_xy1,co_xy2,y1var,y1bar;//covariance, y mean, variance of y
    double y2var,y2bar,co_y1y2;
    double xbar,c1opt,c2opt;

    u = ((double)rand()/(RAND_MAX));
    us[0] = u;
    y1 = u;
    y2 = u*u;
    x = sqrt(1-y2);
    xbar = x;

    y1bar = y1;
    y1var = 0;

    y2bar = y2;
    y2var = 0;

    co_xy1 = 0;
    co_xy2 = 0;
    co_y1y2 = 0;
    for(int i = 1; i < 100; i++)
    {
         u = ((double)rand()/(RAND_MAX));
         us[i] = u;
         y1 = u;
         y2 = u*u;
         x = sqrt(1-y2);
         y1var = y1var*(i - 1)/i  + (y1bar - y1)*(y1bar - y1)/(i + 1);
         y1bar = y1bar + (y1 - y1bar)/(i + 1);
         y2var = y2var*(i - 1)/i  + (y2bar - y2)*(y2bar - y2)/(i + 1);
         y2bar = y2bar + (y2 - y2bar)/(i + 1);

         xbar = xbar + (x - xbar)/(i + 1);
         co_xy1 = co_xy1*(i - 1)/i + (x-xbar)*(y1-y1bar)/i; //recursive covariance
         co_xy2 = co_xy2*(i - 1)/i + (x-xbar)*(y2-y2bar)/i; //recursive covariance
         co_y1y2 = co_y1y2*(i - 1)/i + (y1-y1bar)*(y2-y2bar)/i; //recursive covariance
    }
    c1opt = (co_xy2*co_y1y2-co_xy1*y2var)/( y2var*y1var - co_y1y2*co_y1y2); // c1_opt
    c2opt = (co_xy1*co_y1y2-co_xy2*y1var)/( y2var*y1var - co_y1y2*co_y1y2);//c2_opt

    u2 = us[0]*us[0];
    mean = sqrt(1-u2);
    meanA = mean + c1opt*(us[0]-0.5)+c2opt*(u2 - onethird);
    svs = 0;
    svsA = 0;
    for(int i = 1; i < 100; i++)
    {
         u2 = us[i]*us[i];
         x = sqrt(1-u2);
         y = x + c1opt*(us[i]-0.5)+c2opt*(u2 - onethird);
```

```
        svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i + 1);
        svsA = svsA *(i - 1)/i  + (meanA - y)*(meanA - y)/(i + 1);
        mean = mean + (x - mean)/(i+1);
        meanA = meanA + (y - meanA)/(i+1);

    }

    for(int i = 100;i < n;i++){
        u = ((double)rand()/(RAND_MAX));
        u2 = u*u;
        x = sqrt(1-u2);
        y = x + c1opt*(u-0.5) + c2opt*(u2 - onethird);
        svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i + 1);
        svsA = svsA *(i - 1)/i  + (meanA - y)*(meanA - y)/(i + 1);
        mean = mean + (x - mean)/(i+1);
        meanA = meanA + (y - meanA)/(i+1);
    }
    return make_tuple(mean, meanA, svs,svsA);
}
int main(){
    int N[5]={100,1000,10000,100000,1000000};
    for(int i=0;i<5;i++){
        srand(1);       //set up the same random seed for comparison
        auto Var = Control_uu2(N[i]);
        cout<<"The␣muilticontrol␣variate␣is␣u,␣and␣u^2,␣and␣N␣=␣"<<N[i]<<endl;
        cout<<"Value␣of␣crude␣MC␣estimator␣is␣"<<get<0>(Var)<<endl;
        cout<<"Value␣of␣control␣MC␣estimator␣is␣"<<get<1>(Var)<<endl;
        cout<<"Variance␣of␣crude␣MC␣estimator␣is␣"<<get<2>(Var)<<endl;
        cout<<"Variance␣of␣control␣MC␣estimator␣is␣"<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

2.3.2 Results

```
The muilticontrol variate is u, and u^2, and N = 100
Value of crude MC estimator is 0.75306
Value of control MC estimator is 0.786515
Variance of crude MC estimator is 0.0538266
Variance of control MC estimator is 0.00198585

The muilticontrol variate is u, and u^2, and N = 1000
Value of crude MC estimator is 0.784426
Value of control MC estimator is 0.788538
Variance of crude MC estimator is 0.0459687
Variance of control MC estimator is 0.00169225

The muilticontrol variate is u, and u^2, and N = 10000
Value of crude MC estimator is 0.787352
Value of control MC estimator is 0.785134
Variance of crude MC estimator is 0.0498439
Variance of control MC estimator is 0.00203989

The muilticontrol variate is u, and u^2, and N = 100000
Value of crude MC estimator is 0.785401
Value of control MC estimator is 0.785277
Variance of crude MC estimator is 0.0499881
Variance of control MC estimator is 0.00199286

The muilticontrol variate is u, and u^2, and N = 1000000
Value of crude MC estimator is 0.785405
Value of control MC estimator is 0.785389
Variance of crude MC estimator is 0.0498311
Variance of control MC estimator is 0.00197957
```

3 Question 3

In Homeworks 1, 5, and 7, you were asked to estimate $E[M]$ where M is equal to the number of uniformly distributed on (0,1) random number that must be summed to exceed 1. In other words, for uniformly distributed on (0,1) random variables $U_1, U_2, \cdots$,

$$M = \min\{n : \sum_{i=1}^{n} U_i > 1\}.$$

You observed that it appeared that the expected value was e. Use control variates in a simulation to estimate e when using $N = 10^i$ realizations for $i = 2, \cdots, 6$. Compare the variance of your estimator when using control variates with the variance of your previous results obtained without the use of variance reduction technique.

3.1 Codes

```
#include<iostream>
#include<cmath>
#include <tuple>
using namespace std;
tuple<double,double> numGreatOne(){
    double u,sum,sumA;//u~(0,1), sum of us, sum of antithetics
    double i,j;
    u = ((double)rand()/(RAND_MAX));
    sum = u;
    u = ((double)rand()/(RAND_MAX));
    sum = sum + u;
    j = sum;
    i = 2;
    while( (sum<= 1) ){
        u = ((double)rand()/(RAND_MAX));
        sum = sum + u;
        i = i + 1;
    }
    return make_tuple(i,j);
}
tuple<double,double,double,double> getVars(int n){
    double u, copt;           // u(0,1),
    double x, xs[100], y, ys[100], xA;   // realizations
    double xbar,ybar,yvar,co_xy;
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = numGreatOne();
    x = get<0>(results);
    xs[0] = x;
    y = get<1>(results);
    xbar = x;
    ybar = y;
    yvar = 0;
    co_xy = 0;
    for(int i = 1; i < 100; i++)
    {
         auto results = numGreatOne();
         x = get<0>(results);
         y = get<1>(results);
         xs[i] = x;
         ys[i] = y;
         yvar = yvar*(i - 1)/i  + (ybar - y)*(ybar - y)/(i + 1);
         ybar = ybar + (y - ybar)/(i + 1);
         xbar = xbar + (x - xbar)/(i + 1);
         co_xy = co_xy*(i - 1)/i + (x-xbar)*(y-ybar)/i; //recursive covariance
    }
    copt = -co_xy/yvar; // c_opt

    x = xs[0];
    y = ys[0];
    xA = x + copt*(y-1);
    mean = x;
    meanA = xA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < 100; i++)
    {
        x = xs[i];
        y = ys[i];
        xA = x + copt*(y-1);
```

```
        svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i+1);
        mean = mean + (x-mean)/(i+1);

        svsA = svsA *(i - 1)/i  + (meanA - xA)*(meanA - xA)/(i+1);
        meanA = meanA + (xA-meanA)/(i+1);

    }

    for(int i = 100; i < n; i++)
    {
        auto results = numGreatOne();
        x = double(get<0>(results));
        y = double(get<1>(results));
        xA = x + copt*(y-1);
        svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i+1);
        mean = mean + (x-mean)/(i+1);

        svsA = svsA *(i - 1)/i  + (meanA - xA)*(meanA - xA)/(i+1);
        meanA = meanA + (xA-meanA)/(i+1);
    }
    return make_tuple(mean, meanA, svs,svsA);
}

int main(){
    int N[5]={100,1000,10000,100000,1000000};
    for(int i=0;i<5;i++){
        srand(1);       //set up the same random seed for comparison
        auto Var = getVars(N[i]);
        cout<<"For N = "<<N[i]<<endl;
        cout<<"Value of the M is "<<get<0>(Var)<<endl;
        cout<<"Value of Control M is "<<get<1>(Var)<<endl;
        cout<<"Variance of normal M is "<<get<2>(Var)<<endl;
        cout<<"Variance of Control M is "<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

3.2 Results

```
For N = 100
Value of the M is 2.58
Value of Control M is 2.67541
Variance of normal M is 0.710707
Variance of Control M is 0.276671

For N = 1000
Value of the M is 2.713
Value of Control M is 2.73358
Variance of normal M is 0.785416
Variance of Control M is 0.309521

For N = 10000
Value of the M is 2.727
Value of Control M is 2.72016
Variance of normal M is 0.775749
Variance of Control M is 0.301316

For N = 100000
Value of the M is 2.71767
Value of Control M is 2.71649
Variance of normal M is 0.757607
Variance of Control M is 0.288942

For N = 1000000
Value of the M is 2.71869
Value of Control M is 2.71797
Variance of normal M is 0.76404
Variance of Control M is 0.289633
```

4 Question 4

In Homework 3, 5, and 7, you were asked to continually roll a pair of fair dice until all possible outcomes $2, 3, \cdots, 12$ had occurred at least once and conduct a simulation study to approximate the expected number of dice rolls that are needed. Use control variates in a simulation to estimate the expected number of dice rolls when using $N = 10^i$ realizations for $i = 2, \cdots, 6$. Compare the variance of your estimator when using control variates with the variance of your previous results obtained without the use of a variance reduction technique. You should also try at least two different control variates and compare the variance reduction obtained for each of them.

4.1 Control variate A

4.1.1 Code A

```
#include<iostream>
#include<cmath>
#include <tuple>
using namespace std;

tuple<double,double> twoDices(){
    double u1,u2; //u1,u2~u(0,1)
    int n1,n2;     //n1~{1,2,3,4,5,6}
    int outcome;   //n1+n2
    int outComes[11]  = {2,3,4,5,6,7,8,9,10,11,12}; //outcomes
    int outComeSum;//indicator whether all possible outcomes are shown up
    int const maxSum = 77; // sum of all outcomes
    double i,j;//j is the control variate

    outComeSum = 0;
    i = 0;
    j = 0;
    while((outComeSum != maxSum)){
        u1 = ((double) rand() / (RAND_MAX));
        u2 = ((double) rand() / (RAND_MAX));
        n1 = (int)(u1 * 6.0) + 1 ;
        n2 = (int)(u2 * 6.0) + 1 ;
        outcome = n1 + n2;

        if( (outcome==12) &&(j==0)) {
            j = i + 1;
        }

        if( outcome == outComes[outcome-2]){
            outComes[outcome-2] = 0;
            outComeSum = outComeSum + outcome;
        }
        i = i + 1;
    }
    return make_tuple(i,j);
}
tuple<double,double,double,double> getVars(int n){
    double u, copt;          // u(0,1),
    double x, xs[100], y, ys[100], xA;   // realizations
    double xbar,ybar,yvar,co_xy;
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = twoDices();
    x = get<0>(results);
    xs[0] = x;
    y = get<1>(results);
    ys[0] = y;

    xbar = x;
    ybar = y;
    yvar = 0;
    co_xy = 0;
    for(int i = 1; i < 100; i++)
    {
         auto results = twoDices();
         x = get<0>(results);
         y = get<1>(results);
         xs[i] = x;
         ys[i] = y;
         yvar = yvar*(i - 1)/i  + (ybar - y)*(ybar - y)/(i + 1);
         ybar = ybar + (y - ybar)/(i + 1);
```

```
            xbar = xbar + (x - xbar)/(i + 1);
            co_xy = co_xy*i/(i+1) + (x-xbar)*(y-ybar)/(i+1); //recursive covariance
        }
        copt = -co_xy/yvar; // c_opt

        x = xs[0];
        y = ys[0];
        xA = x + copt*(y-36);
        mean = x;
        meanA = xA;
        svs = 0;
        svsA = 0;

        for(int i = 1; i < 100; i++)
        {
            x = xs[i];
            y = ys[i];
            xA = x + copt*(y-36);
            svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i+1);
            mean = mean + (x-mean)/(i+1);
            svsA = svsA *(i - 1)/i  + (meanA - xA)*(meanA - xA)/(i+1);
            meanA = meanA + (xA-meanA)/(i+1);
        }

        for(int i = 100; i < n; i++)
        {
            auto results = twoDices();
            x = double(get<0>(results));
            y = double(get<1>(results));
            xA = x + copt*(y-36);
            svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i+1);
            mean = mean + (x-mean)/(i+1);

            svsA = svsA *(i - 1)/i  + (meanA - xA)*(meanA - xA)/(i+1);
            meanA = meanA + (xA-meanA)/(i+1);
        }
        return make_tuple(mean, meanA, svs,svsA);
}

int main(){
        int N[5]={100,1000,10000,100000,1000000};
        for(int i=0;i<5;i++){
            srand(4);       //set up the same random seed for comparison
            auto Var = getVars(N[i]);
            cout<<"For N = "<<N[i]<<endl;
            cout<<"Value of the crude M is "<<get<0>(Var)<<endl;
            cout<<"Value of Control M is "<<get<1>(Var)<<endl;
            cout<<"Variance of the crude M is "<<get<2>(Var)<<endl;
            cout<<"Variance of Control M is "<<get<3>(Var)<<endl;
            cout<<endl;
        }
}
```

4.1.2 Result A

```
For N = 100
Value of the crude M is 56.45
Value of Control M is 59.9729
Variance of the crude M is 1055.89
Variance of Control M is 606.477

For N = 1000
Value of the crude M is 60.332
Value of Control M is 59.6738
Variance of the crude M is 1227.75
Variance of Control M is 624.309

For N = 10000
Value of the crude M is 61.6402
Value of Control M is 61.4976
Variance of the crude M is 1372.36
Variance of Control M is 819.163

For N = 100000
Value of the crude M is 61.2787
```

```
Value of Control M is 61.3162
Variance of the crude M is 1294.98
Variance of Control M is 805.463

For N = 1000000
Value of the crude M is 61.1476
Value of Control M is 61.2278
Variance of the crude M is 1283.03
Variance of Control M is 800.609
```

4.2 Control variate B

4.2.1 Code B

```
#include<iostream>
#include<cmath>
#include <tuple>
using namespace std;

tuple<double,double> twoDices(){
    double u1,u2; //u1,u2~u(0,1)
    int n1,n2;     //n1~{1,2,3,4,5,6}
    int outcome;   //n1+n2
    int outComes[11]  = {2,3,4,5,6,7,8,9,10,11,12}; //outcomes
    int outComeSum;//indicator whether all possible outcomes are shown up
    int const maxSum = 77; // sum of all outcomes
    double i,j,k;//j is the control variate

    outComeSum = 0;
    i = 0;
    j = 0;
    k = 0;
    while((outComeSum != maxSum)){
        u1 = ((double) rand() / (RAND_MAX));
        u2 = ((double) rand() / (RAND_MAX));
        n1 = (int)(u1 * 6.0) + 1 ;
        n2 = (int)(u2 * 6.0) + 1 ;
        outcome = n1 + n2;

        if( (outcome == 2 )&&(j==0)){
            j = i + 1;
        }
        if( (outcome == 12 )&&(k==0)){
            k = i + 1;
        }
        if( outcome == outComes[outcome-2]){
            outComes[outcome-2] = 0;
            outComeSum = outComeSum + outcome;
        }
        i = i + 1;
    }
    j = j + k;
    return make_tuple(i,j);
}
tuple<double,double,double,double> getVars(int n){
    double u, copt;          // u(0,1),
    double x, xs[100], y, ys[100], xA;   // realizations
    double xbar,ybar,yvar,co_xy;
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = twoDices();
    x = get<0>(results);
    xs[0] = x;
    y = get<1>(results);
    ys[0] = y;

    xbar = x;
    ybar = y;
    yvar = 0;
    co_xy = 0;
    for(int i = 1; i < 100; i++)
    {
         auto results = twoDices();
         x = get<0>(results);
         y = get<1>(results);
```

```
            xs[i] = x;
            ys[i] = y;
            yvar = yvar*(i - 1)/i  + (ybar - y)*(ybar - y)/(i + 1);
            ybar = ybar + (y - ybar)/(i + 1);
            xbar = xbar + (x - xbar)/(i + 1);
            co_xy = co_xy*i/(i+1) + (x-xbar)*(y-ybar)/(i+1); //recursive covariance
        }
        copt = -co_xy/yvar; // c_opt

        x = xs[0];
        y = ys[0];
        xA = x + copt*(y-72);
        mean = x;
        meanA = xA;
        svs = 0;
        svsA = 0;

        for(int i = 1; i < 100; i++)
        {
            x = xs[i];
            y = ys[i];
            xA = x + copt*(y-72);
            svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i+1);
            mean = mean + (x-mean)/(i+1);
            svsA = svsA *(i - 1)/i  + (meanA - xA)*(meanA - xA)/(i+1);
            meanA = meanA + (xA-meanA)/(i+1);
        }

        for(int i = 100; i < n; i++)
        {
            auto results = twoDices();
            x = double(get<0>(results));
            y = double(get<1>(results));
            xA = x + copt*(y-72);
            svs = svs *(i - 1)/i  + (mean - x)*(mean - x)/(i+1);
            mean = mean + (x-mean)/(i+1);

            svsA = svsA *(i - 1)/i  + (meanA - xA)*(meanA - xA)/(i+1);
            meanA = meanA + (xA-meanA)/(i+1);
        }
        return make_tuple(mean, meanA, svs,svsA);
}

int main(){
        int N[5]={100,1000,10000,100000,1000000};
        for(int i=0;i<5;i++){
            srand(4);       //set up the same random seed for comparison
            auto Var = getVars(N[i]);
            cout<<"For␣N␣=␣"<<N[i]<<endl;
            cout<<"Value␣of␣the␣crude␣M␣is␣"<<get<0>(Var)<<endl;
            cout<<"Value␣of␣Control␣M␣is␣"<<get<1>(Var)<<endl;
            cout<<"Variance␣of␣the␣crude␣M␣is␣"<<get<2>(Var)<<endl;
            cout<<"Variance␣of␣Control␣M␣is␣"<<get<3>(Var)<<endl;
            cout<<endl;
        }
}
```

4.2.2 Result B

```
For N = 100
Value of the crude M is 56.45
Value of Control M is 59.8517
Variance of the crude M is 1055.89
Variance of Control M is 278.942

For N = 1000
Value of the crude M is 60.332
Value of Control M is 60.6381
Variance of the crude M is 1227.75
Variance of Control M is 297.81

For N = 10000
Value of the crude M is 61.6402
Value of Control M is 61.562
Variance of the crude M is 1372.36
```

```
Variance of Control M is 321.093

For N = 100000
Value of the crude M is 61.2787
Value of Control M is 61.3363
Variance of the crude M is 1294.98
Variance of Control M is 307.322

For N = 1000000
Value of the crude M is 61.1476
Value of Control M is 61.2926
Variance of the crude M is 1283.03
Variance of Control M is 307.232
```