Assignment 9

Chao Cheng

April 16, 2019

1 Question 1

Suppose that one wishes to use simulation to estimate $P\{X+Y>4\}$ where X and Y are independent exponentials with parameters 1 and 1/2 respectively. Use conditional Monte Carlo in a simulation study of this probability. Compare your results to those from a Monte Carlo simulation in which no variance reduction technique was used and to those from a Monte Carlo simulation in which either antithetic variates or control variates were used.

1.1 Conditioning MC

1.2 Code

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

tuple<double,double> twoDices(){
    double u1,u2; //u1,u2~u(0,1)
    double e1,e2; //exp(1),exp(0.5)
    double i,j;//
    u1 = ((double)rand()/(RAND_MAX));
    u2 = ((double)rand()/(RAND_MAX));
    e1 = -log(u1);
    e2 = -2*log(u2);
    i = 0;
    j = 0;
    if ((e1+e2>4)) {
        i = 1.0;
    }
    if (e2>=4.0) {
        j = 1.0;
    }else{
        j = exp(e2 - 4.0);
    }
    return make_tuple(i,j);
}
tuple<double,double,double,double> getVars(int n){
    double u, copt;          // u(0,1),
    double x, y, xA;   // realizations
    double xbar,ybar,yvar;
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = twoDices();
    x = get<0>(results);
    y = get<1>(results);
    xA = y;
    mean = x;
    meanA = xA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < n; i++)
    {
        auto results = twoDices();
        x = double(get<0>(results));
        y = double(get<1>(results));
        xA = y;
        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 MC is "<<get<0>(Var)<<endl;
        cout<<"Value of Conditioning MC is "<<get<1>(Var)<<endl;
        cout<<"Variance of the crude MC is "<<get<2>(Var)<<endl;
        cout<<"Variance of Conditioning MC is "<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

1.2.1 Results

```
For N = 100
Value of the crude MC is 0.27
Value of Conditioning MC is 0.266878
Variance of the crude MC is 0.199091
Variance of Conditioning MC is 0.125768

For N = 1000
Value of the crude MC is 0.263
Value of Conditioning MC is 0.255876
Variance of the crude MC is 0.194025
Variance of Conditioning MC is 0.117664

For N = 10000
Value of the crude MC is 0.2568
Value of Conditioning MC is 0.253959
Variance of the crude MC is 0.190873
Variance of Conditioning MC is 0.116809

For N = 100000
Value of the crude MC is 0.25071
Value of Conditioning MC is 0.251934
Variance of the crude MC is 0.187856
Variance of Conditioning MC is 0.116131

For N = 1000000
Value of the crude MC is 0.252026
Value of Conditioning MC is 0.252263
Variance of the crude MC is 0.188509
Variance of Conditioning MC is 0.116557
```

1.3 Antithetic MC

1.3.1 Code

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

tuple<double,double> twoDices(){
    double u1,u2; //u1,u2~u(0,1)
    double e1,e2; //exp(1),exp(0.5)
    double i,j;//
    u1 = ((double)rand()/(RAND_MAX));
    u2 = ((double)rand()/(RAND_MAX));
    e1 = -log(u1);
    e2 = -2*log(u2);
    i = 0;
    j = 0;
    if ((e1+e2>4)) {
        i = 1.0;
    }
    if (-log(1-u1)-2*log(1-u2)>4) {
        j = 1.0;
    }
    return make_tuple(i,j);
}
tuple<double,double,double,double> getVars(int n){
```

```
    double u, copt;           // u(0,1),
    double x, y, xA;   // realizations
    double xbar,ybar,yvar;
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = twoDices();
    x = get<0>(results);
    y = get<1>(results);
    xA = (x+y)/2.0;
    mean = x;
    meanA = xA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < n; i++)
    {
        auto results = twoDices();
        x = double(get<0>(results));
        y = double(get<1>(results));
        xA = (x+y)/2.0;
        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 MC is "<<get<0>(Var)<<endl;
        cout<<"Value of Antithetic MC is "<<get<1>(Var)<<endl;
        cout<<"Variance of the crude MC is "<<get<2>(Var)<<endl;
        cout<<"Variance of Antithetic MC is "<<get<3>(Var)<<endl;
        cout<<endl;
    }
}
```

1.3.2 Results

```
For N = 100
Value of the crude MC is 0.27
Value of Antithetic MC is 0.24
Variance of the crude MC is 0.199091
Variance of Antithetic MC is 0.0630303

For N = 1000
Value of the crude MC is 0.263
Value of Antithetic MC is 0.2515
Variance of the crude MC is 0.194025
Variance of Antithetic MC is 0.0650628

For N = 10000
Value of the crude MC is 0.2568
Value of Antithetic MC is 0.25535
Variance of the crude MC is 0.190873
Variance of Antithetic MC is 0.0653779

For N = 100000
Value of the crude MC is 0.25071
Value of Antithetic MC is 0.252095
Variance of the crude MC is 0.187856
Variance of Antithetic MC is 0.0654813

For N = 1000000
Value of the crude MC is 0.252026
Value of Antithetic MC is 0.252022
Variance of the crude MC is 0.188509
Variance of Antithetic MC is 0.065389
```

2 Question 2

To better understand the impact of stratified sampling on the simulation of nonuniform random variables, produce histograms similar to those appearing in the textbook in FIgure 4.5 for the simulation of exponentially distributed random variables with parameter 1. Experiment using several different numbers of strata and numbers of samples per strata and numbers of samples per strata.

2.1 Simulation Code

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

int main(){
    // 50,and 100 strata for this experiment
    // 10, and 5 samples per strata accordingly
    double u,u_; //u~u(0,1)
    double p1[100]  = {0};//5 sample
    double p2[50] = {0};// 10 sample

    ofstream strata50;
    ofstream strata100;
    strata50.open("problem2a.csv");
    strata100.open("problem2b.csv");

    for(int i = 0; i < 100; i++)//equiprobable
    {
        p1[i] = ((double)0.01*i);
    }

    for(int i = 0; i < 50; i++)
    {
        p2[i] = ((double)0.02*i);
    }
    for(int i = 0; i < 100; i++)
    {
        for(int j = 0; j < 5; j++)
        {
            u = ((double)rand()/(RAND_MAX));
            u_ = 0.01 * u + p1[i];
            strata100 << -log(u_)<<"\n";
        }
    }
    for(int i = 0; i < 50; i++)
    {
        for(int j = 0; j < 10; j++)
        {
            u = ((double)rand()/(RAND_MAX));
            u_ = 0.02 * u + p2[i];
            strata50 << -log(u_) <<"\n";
        }
    }
    strata100.close();
    strata50.close();
    return 0;
}
```

2.2 Plotting Code

```
load problem2a.csv;
figure();
subplot(1,2,1)
h = histogram(problem2a,'Normalization','pdf');
hold on;
X=linspace(0.001,10.0,1000);
f = @(x)(exp(-x));
plot(X,f(X),'r');
legend('Histogram␣of␣simulated␣values',Exponential distribution with parameter 1')
title("50␣strata,␣10␣samples␣per␣strata")
hold␣off;

load␣problem2b.csv
subplot(122)
h␣=␣histogram(problem2b,'Normalization','pdf');
```

```
hold␣on;
X=linspace(0.001,10.0,1000);
f␣=␣@(x)(exp(-x));
plot(X,f(X),'r');
legend('Histogram of simulated values','Expential distribution with parameter 1')
title("100␣strata,␣5␣samples␣per␣strata")
```

2.3 Results

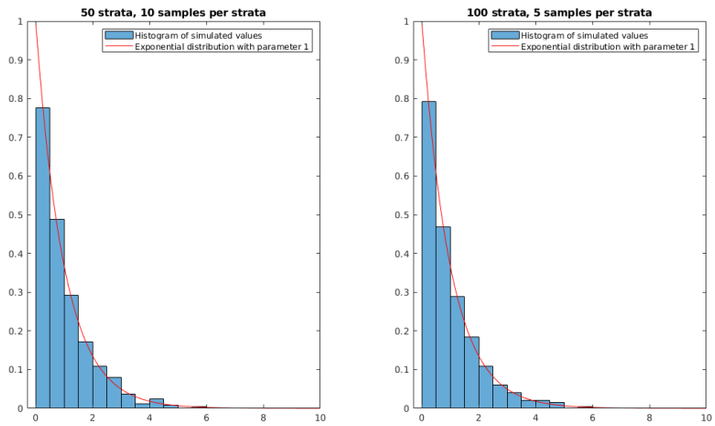

Figure 1: Comparison between different strata and sampling

3 Question 3

In Homeworks 5,7 and 8, you were asked to use Monte Carlo integration to approximate

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

Use stratified sampling in a simulation to approximate this integral. Compare the variance of your estimator when using stratified sampling with the variance of your previous results obtained without the use of a variance reduction technique.

3.1 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> MCIntegrel(){
    double u, u_;    // u(0,1)
    double x, y;    //realization of exp(u^2)
    double est,estA;   //est.
    int n = 1000;//1000 points

    for(int i = 0;i < n;i++){
         u = ((double)rand()/(RAND_MAX));
         u_ = ((double) (u + i) / n);
         x = exp(u*u);
         y = exp(u_*u_);
         est = est + (x - est)/(i+1);
         estA = estA + (y - estA)/(i+1);
    }
    return make_tuple(est,estA);
}
tuple<double,double,double,double> getVars(int n){
    double u;           // u(0,1),
    double x, y, xA;   // realizations
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = MCIntegrel();
    x = double(get<0>(results));
    y = double(get<1>(results));
    xA = y;
    mean = x;
    meanA = xA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < n; i++)
    {
        auto results = MCIntegrel();
        x = double(get<0>(results));
        y = double(get<1>(results));
        xA = y;
        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=1000;
    srand(1);       //set up the same random seed for comparison
    auto Var = getVars(N);
    cout<<"For N = "<<N<<endl;
    cout<<"Mean Value of crude MC integrel is "<<get<0>(Var)<<endl;
    cout<<"Mean Value of Stratified Sampling MC Integrel is "<<get<1>(Var)<<endl;
    cout<<"Variance of crude MC Integrel is "<<get<2>(Var)<<endl;
    cout<<"Variance of Stratified Sampling MC Integrel is "<<get<3>(Var)<<endl;
    cout<<endl;
}
```

3.2 Results

```
For N = 1000
Mean Value of crude MC integrel is 1.46261
Mean Value of Stratified Sampling MC Integrel is 1.46265
Variance of crude MC Integrel is 0.00024643
Variance of Stratified Sampling MC Integrel is 4.32473e-10
```

4 Question 4

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 stratified sampling 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.

4.1 Code

```
#include<iostream>
#include<cmath>
#include <tuple>
using namespace std;
double numGreatOne(){
    double u,sum;//u~(0,1), sum of us, sum of antithetics
    double i;
    u = ((double)rand()/(RAND_MAX));
    i = 1;
    sum = u;
    while( (sum<= 1) ){
        u = ((double)rand()/(RAND_MAX));
        sum = sum + u;
        i = i + 1;
    }
    return i;
}
double numGreatOneStrata(int I, int N){
    double u,u_,sum;//u~(0,1), sum of us, sum of antithetics
    double i;
    u = ((double)rand()/(RAND_MAX));
    u_ = ((double)(u + I)/N);
    i = 1;
    sum = u_;
    while(sum<=1){
        u = ((double)rand()/(RAND_MAX));
        sum = sum + u;
        i = i + 1;
    }
    return i;
}
tuple<double, double>estimator(){
    int n = 10000; // 10000 realizations to estimate
    int strata = 100;// 100 strata
    double x, xs[strata],xA;//Crude estimator, Stratified sampling estimator
    double est, estA,tmp;

    est = 0;
    estA = 0;
    for(int i = 0; i < 10000; i++)
    {
        x = numGreatOne();
        est = est + (x-est)/(i+1);
    }

    for(int i = 0; i < strata; i++){
        tmp = 0;
        for(int j = 0; j < 100; j++)// 100 realization per strata
        {
            xA = numGreatOneStrata(i,strata);
            tmp = tmp + (xA-tmp)/(j+1);
        }
        estA = estA + (tmp - estA)/(i + 1);//since pi is the same, as pi=1/100
    }
    return make_tuple(est,estA);
}

tuple<double,double,double,double> getVars(int n){
    double u;           // u(0,1),
    double x, y, xA;    // realizations
    double mean, svs;   //mean, sample variance of estimator
```

```
    double meanA,svsA; //mean, var of antithetic

    auto results = estimator();
    x = double(get<0>(results));
    y = double(get<1>(results));
    xA = y;
    mean = x;
    meanA = xA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < n; i++)
    {
        auto results = estimator();
        x = double(get<0>(results));
        y = double(get<1>(results));
        xA = y;
        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=100; //generate 100 estimator
    srand(1);       //set up the same random seed for comparison
    auto Var = getVars(N);
    cout<<"For N = "<<N<<endl;
    cout<<"Mean Value of crude MC Estimator is "<<get<0>(Var)<<endl;
    cout<<"Mean Value of Stratified Sampling MC Estimator is "<<get<1>(Var)<<endl;
    cout<<"Variance of crude MC Estimator is "<<get<2>(Var)<<endl;
    cout<<"Variance of Stratified Sampling MC Estimator is "<<get<3>(Var)<<endl;
    cout<<endl;
}
```

4.2 Result

```
For N = 100
Mean Value of crude MC Estimator is 2.7172
Mean Value of Stratified Sampling MC Estimator is 2.71809
Variance of crude MC Estimator is 8.81383e-05
Variance of Stratified Sampling MC Estimator is 5.25547e-05
```

5 Question 5

In Homework 3, 5, 7, and 8, 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 stratified sampling in a simulation to estimate the expected number of dice rolls. Compare the variance of your estimator when using stratified sampling with the variance of your previous results obtained without the use of a variance reduction technique.

5.1 Code

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

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;//
    outComeSum = 0;
    i = 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 == outComes[outcome-2]){
            outComes[outcome-2] = 0;
            outComeSum = outComeSum + outcome;
        }
        i = i + 1;
    }
    return i;
}

double twoDicesStrata(int I){
    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;//

    outComeSum = I + 2;
    outComes[I] = 0;
    i = 1;
    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 == outComes[outcome-2]){
            outComes[outcome-2] = 0;
            outComeSum = outComeSum + outcome;
        }
        i = i + 1;
    }
    return i;
}

tuple<double, double>estimator(){
    int n = 36000; // 11000 realizations to estimate
    int strata = 11;// 11 strata
    double x, xs[strata],xA;//Crude estimator, Stratified sampling estimator
    double est, estA, tmp;
    double ps[11] = {1.0/36,2.0/36,3.0/36,4.0/36,5.0/36,6.0/36,5.0/36,4.0/36,3.0/36,2.0/3
    double ni[11] = {1000,2000,3000,4000,5000,6000,5000,4000,3000,2000,1000};//n*pi
```

```
    est = 0;
    for(int i = 0; i < 10000; i++)
    {
        x = twoDices();
        est = est + (x-est)/(i+1);
    }
    estA = 0;
    for(int i = 0; i < strata; i++){
        tmp = 0;
        for(int j = 0; j < ni[i]; j++)// 1000 realization per strata
        {
            xA = twoDicesStrata(i);
            tmp = tmp + (xA - tmp)/(j + 1);
        }
        estA = estA + tmp*ps[i];//since pi is the same, as pi=1/100
    }
    return make_tuple(est, estA);
}

tuple<double,double,double,double> getVars(int n){
    double u;           // u(0,1),
    double x, xA;   // realizations
    double mean, svs;  //mean, sample variance of estimator
    double meanA,svsA; //mean, var of antithetic

    auto results = estimator();
    x = double(get<0>(results));
    xA = double(get<1>(results));
    mean = x;
    meanA = xA;
    svs = 0;
    svsA = 0;
    for(int i = 1; i < n; i++)
    {
        auto results = estimator();
        x = double(get<0>(results));
        xA = double(get<1>(results));
        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 = 100;    //generate 100 value of estimators
    srand(1);     //set up the same random seed for comparison
    auto Var = getVars(N);
    cout<<"For␣N␣=␣"<<N<<"␣realizations␣of␣estimators"<<endl;
    cout<<"Each␣estimator␣is␣generated␣from␣36000␣realizations"<<endl;
    cout<<"Mean␣Value␣of␣crude␣MC␣Estimator␣is␣"<<get<0>(Var)<<endl;
    cout<<"Mean␣Value␣of␣Stratified␣Sampling␣MC␣Estimator␣is␣"<<get<1>(Var)<<endl;
    cout<<"Variance␣of␣crude␣MC␣Estimator␣is␣"<<get<2>(Var)<<endl;
    cout<<"Variance␣of␣Stratified␣Sampling␣MC␣Estimator␣is␣"<<get<3>(Var)<<endl;
    cout<<endl;
}
}
```

5.2 Result

```
For N = 100 realizations of estimators
Each estimator is generated from 36000 realizations
Mean Value of crude MC Estimator is 61.1615
Mean Value of Stratified Sampling MC Estimator is 61.1889
Variance of crude MC Estimator is 0.124573
Variance of Stratified Sampling MC Estimator is 0.0399827
```