FIM 548
Homework - 3

Venkata Sachin Chandra Margam

22 March 2023

Contents

1 Problem (Control Variates and Antithetic Sampling) **2**

2 Problem (Conditional Monte Carlo) **8**

3 Problem (Quasi Monte Carlo) **18**

1 Problem (Control Variates and Antithetic Sampling)

Let the underlying price have risk-neutral SDE

$$\frac{dS_t}{S_t} = (r - q)dt + \sigma dW_t.$$

An Asian option with maturity T and strike K has payoff

$$\left(\frac{1}{m}\sum_{i=1}^{m} S_{t_i} - K\right)^+,$$

where $t_i = iT/m$. We want to evaluate the price of this option for $T = 1, m = 4, S_0 = 100, r = 0.05, q = 0.02, \sigma = 0.2$, and for range of strikes $K = 90, 95, 100, 105, 110, 115, 120$.
p

1. **Use the standard MC method to price the contracts, using N = 100 sample paths. Generate 100 of these MC estimators and save in a $7 * 100$ array.**

   Solution:

```
%% Asian Option Control Variates
S0 = 100;
sigma = .2;
r = .05;
q = .02;
T = 1;
m = 4;
K = 90:5:120;
D = exp(-r*T);
t = linspace(T/m,T,m);

%% compare standard MC to Control Variates
N = 100;  %% Monte Carlo sample size
L = 100;  %% number of trials
C_mc = zeros(length(K),L);
C_cv = zeros(length(K),L);
for ctr = 1:L
    %%%% Monte Carlo Asian Call
    W = cumsum(randn(N,m)*sqrt(T/m),2);
    S = S0*exp((r-q-.5*sigma^2)*t + sigma*W);
    M = mean(S,2);
    [Mv,Kv] = meshgrid(M,K);
    psi_asian = D*max(Mv-Kv,0);
    C_mc(:,ctr) = mean(psi_asian,2);
end
%% compute mean and stnd err of trials
avg = [mean(C_mc,2)];
err = [std(C_mc,[],2)];
Avg_Prices = array2table(avg,'VariableNames', {'Avg-Std-MC'});
Avg_err = array2table(err,'VariableNames', {'Std-err-Std-MC'});
```

2. **Use the control variates method to price the contracts, using $N = 100$ sample paths. For the auxiliary payoff use the geometric mean: $\left(\left(\prod_{i=1}^{m} S_{t_i}\right)^{1/m} - K\right)^{+}$, for which there is an explicit expression for the expected value. Generate 100 of these MC estimators and save in a $7 * 100$ array.**

   Solution:

```
%% Asian Option Control Variates
S0 = 100;
sigma = .2;
r = .05;
q = .02;
T = 1;
m = 4;
K = 90:5:120;
D = exp(-r*T);
t = linspace(T/m,T,m);
%% compare standard MC to Control Variates
N = 100;  %% Monte Carlo sample size
L = 100;  %% number of trials
C_mc = zeros(length(K),L);
C_cv = zeros(length(K),L);
for ctr=1:L
    W = cumsum(randn(N,m)*sqrt(T/m),2);
    S = S0*exp((r-q-.5*sigma^2)*t + sigma*W);
    M = mean(S,2);
    [Mv,Kv] = meshgrid(M,K);
    psi_asian = D*max(Mv-Kv,0);
    C_mc(:,ctr) = mean(psi_asian,2);
    for i = 1:m
        Mmt1(i) = exp((((r-q)-.5*sigma^2)*((T*i)/m^2))+...
        ((sigma^2*T*i*i)/(2*m^3)));
        Mmt2(i) = exp((2*((r-q)-.5*sigma^2)*((T*i)/(m^2)))+...
        (((2*sigma^2*T)/m^3)*(i*i)));
    end
    Mmt1_new = prod(Mmt1);
    Mmt2_new = prod(Mmt2);
    ST = S0*Mmt1_new;
    sig_new = sqrt(log(Mmt2_new/(Mmt1_new^2))/T);
    %%%% control variates with Standard Call
    G = geomean(S,2);
    [Gv,Kv] = meshgrid(G,K);
    psi_geoasian = D*max(Gv-Kv,0);
    Epsi_geoasian = blkprice(ST,K,r-q,T,sig_new);
    b = (psi_asian - mean(psi_asian,2))*(psi_geoasian - Epsi_geoasian')'/N;
    b = diag(b)./diag((psi_geoasian-Epsi_geoasian')*...
    (psi_geoasian-Epsi_geoasian')'/N);
    psi_cv = psi_asian - diag(b)*(psi_geoasian - Epsi_geoasian');
    C_cv(:,ctr) = mean(psi_cv,2);
end
%% compute mean and stnd err of trials
avg = [mean(C_mc,2),mean(C_cv,2)];
err = [std(C_mc,[],2),std(C_cv,[],2)];
Avg_Prices = array2table(avg,'VariableNames', ...
{'Avg-Std-MC','Avg-ControlVariate'});
std_err = array2table(err,'VariableNames', ...
{'Std-err-Std-MC','Std-err-ControlVariate'});
```

3. **Use antithetic sampling $\widetilde{W} = -W$ to price the contracts, using antithetic sample size $N = 50$. Generate 100 of these MC estimators and save in a $7 * 100$ array.**

Solution:

```
%Asian Option Control Variates
S0 = 100;
sigma = .2;
r = .05;
q = .02;
T = 1;
m = 4;
K = 90:5:120;
D = exp(-r*T);
t = linspace(T/m,T,m);

%compare standard MC to Control Variates
N = 50;  %% Monte Carlo sample size
L = 100;  %% number of trials
C_mc = zeros(length(K),L);
C_cv = zeros(length(K),L);
for ctr = 1:L
    %%% Monte Carlo Asian Call
    W = cumsum(randn(N,m)*sqrt(T/m),2);
    S = S0*exp((r-q-.5*sigma^2)*t + sigma*W);
    M=mean(S,2);
    [Mv,Kv] = meshgrid(M,K);
    psi_asian = 0.5*D*max(Mv-Kv,0);
    C_mc(:,ctr) = mean(psi_asian,2);

    %%% antithetic variates with Standard Call
    tildeS=S0*exp((r-q-.5*sigma^2)*t - sigma*W);
    M1=mean(tildeS,2);
    [Mv1,Kv] = meshgrid(M1,K);
    tilde_psi_euro = 0.5*D*max(Mv1-Kv,0);
    C_ant(:,ctr)=mean((psi_asian+tilde_psi_euro),2);
end

%compute mean and stnd err of trials
avg = [mean(C_ant,2)];
err = [std(C_ant,[],2)];

Avg_Prices = array2table(avg,'VariableNames', {'Avg-Antithetic'});
std_err = array2table(err,'VariableNames', {'Std-err-Antithetic'});
```

On a single pair of axes, plot the average of estimators from parts 1, 2, and 3 with K along the horizontal axis. Compute the standard error for each of the estimations in parts 1, 2, and 3, and fill in Table 1. Comment on any reduction in variance that is seen in the table.

```
figure
plot(K,mean(C_mc,2))
title('Option Prices')
xlabel('K-Values')
ylabel('Asian Option Prices')

hold on

plot(K,mean(C_cv,2))
plot(K,mean(C_ant,2))
legend('Standard MC','Control Variate','Antithetic')

hold off
```

Standard Errors of Asian Option Prices			
Strike	MC	Ctrl Var	Antithetic
90	1.1529	0.0524	0.5536
95	1.0264	0.0446	0.6606
100	0.8734	0.0389	0.7188
105	0.6931	0.0340	0.6917
110	0.5465	0.0298	0.6011
115	0.4254	0.0273	0.4873
120	0.3296	0.0283	0.3670

Figure 1: Table with standard errors

Output:

Looking at the table above, we can see that the standard errors of the control variates method are significantly smaller than those of the standard MC and antithetic sampling methods. The standard errors of the antithetic sampling method are smaller than those of the standard MC method, but they are still larger than those of the control variates method. This suggests that the control variates method has reduced the variance of the estimator to a greater level compared to the other two methods.

Control variates method is a powerful variance reduction technique that works by adding a second, known, function to the original function whose value is to be estimated. This second function should be correlated with the original function, and its value at the true parameter value should be known. By using the known value of the second function, the variance of the original function can be reduced. Whereas antithetic sampling works by using correlated random variables. Specifically, antithetic sampling generates pairs of random variables, one of which is the negative of the other. By using these pairs of variables, the variance of the estimator can be reduced.

Overall, the control variates method appears to be the most effective at reducing the variance of the estimator, followed by the antithetic sampling method. The control variates method can be more effective at reducing variance than antithetic sampling when the auxiliary function is chosen carefully. However, the control variates method can be computationally more expensive than antithetic sampling, especially when the auxiliary function is not easily available or requires significant computation.

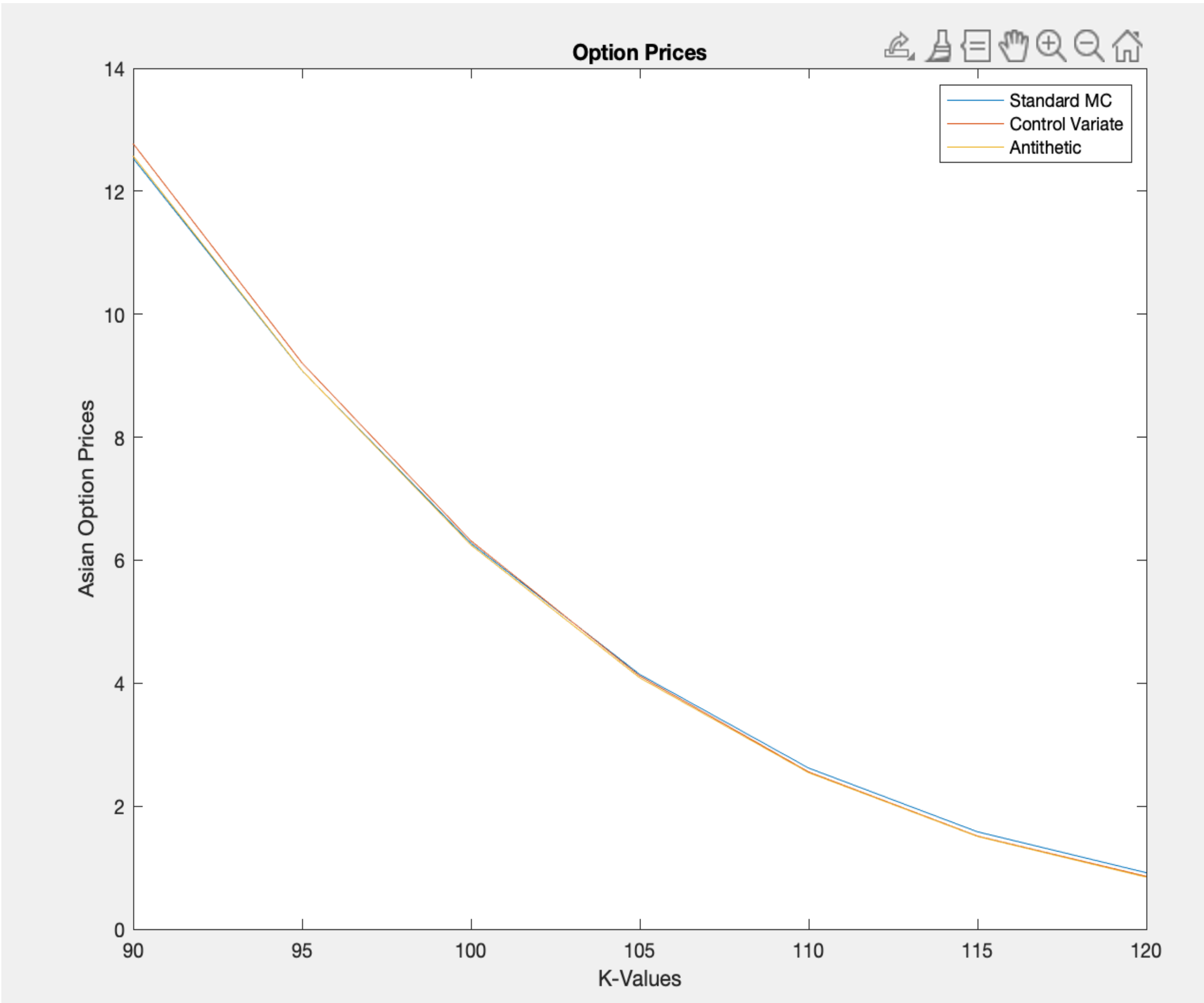

Figure 2: Graph

2 Problem (Conditional Monte Carlo)

Assume that the stock price follows Heston stochastic volatility model under a risk-neutral measure,

$$\frac{dS_t}{S_t} = rdt + \sqrt{X_t}\left(\rho dB_t + \sqrt{1-\rho^2}dW_t\right)$$

$$dX_t = \kappa(\theta - X_t)dt + \sigma\sqrt{X_t}dB_t,$$

where B and W are independent standard Brownian motions. Use 20,000 Monte Carlo samples to price a European call option with $S_0 = 100$, $r = 0.05$, $T = 3/12$ and strikes $K = 90, 95, 100, 105, 110, 115, 120$. Take the CIR parameters to be $X_0 = 0.2$, $\kappa = 3$, $\theta = 0.2$, $\sigma = \sqrt{2\theta\kappa}$, and we'll adjust ρ. In simulating the SDEs take a step size of $\Delta t = 1/365$ (i.e., 90 steps). Perform the following analysis separately for $\rho = 0, -0.3, -0.7$:

Finally, explain why there might be more variance reduction when $\rho = 0$ and less when $|\rho|$ increases.

1. **Estimate the call price using standard MC. Generate 100 MC estimators and save in a $7 * 100$ array. Also save each estimator's implied volatility in a $7 * 100$ array.**

   Solution:

```
clear
clc
X0 = 0.20;
S0 = 100;
kappa = 3;
theta = 0.2;
sigma = sqrt(2*theta*kappa);
T = 3/12;
r = 0.05;
rhos = [0, -0.3, -0.7];
K_values = (90:5:120);
plot_num = 1;
M = 20000;
N = round(T*365);
dt = T/N;
T = 3/12;
n = round(365*T);
steps = floor(T/dt);
L = 100;
call_price=zeros(M,length(rhos),length(K_values));
bls=zeros(L,length(rhos),length(K_values));
call=zeros(L,length(rhos),length(K_values));

for ctr =1:L
    for j=1:length(rhos)
        rho = rhos(j);
        X = X0*ones(M, steps);
        X(:, 1) = X0;
        S = log(S0)*ones(M, steps);
        for t = 2:steps
            dW = randn(M, 1) .* sqrt(dt);
            dB = randn(M, 1) .* sqrt(dt);
            X(:, t) = (1 - kappa*dt) .* X(:, t-1) + kappa*theta*dt + ...
            sigma .* sqrt(max(X(:, t-1), 0)) .* dW;
            S(:, t) = S(:, t-1) + (r - 0.5 * X(:, t-1))*dt + ...
            sqrt(max(X(:, t-1), 0)).*(rho*dW + sqrt(1-rho^2)*dB);
        end
        S = exp(S);
        S_T = S(:, end);
        [Sv, Kv] = meshgrid(S_T, K_values);
        payoffs = max(Sv - Kv, 0);
        call_price(:,j,:) = payoffs' .* exp(-r*T);
    end
    call(ctr,:,:) = mean(call_price,1);
    for k=1:length(K_values)
        bls(ctr,:,k)= blsimpv(S0, K_values(k), r, T, call(ctr,:,k));
    end
end
```

```
call_final = mean(call,1);
bls_final = mean(bls,1);

C = permute(call_final,[1 3 2]);
C = reshape(C,[],size(call_final,2),1);
Avg_Prices_Table = array2table(C,'VariableNames', ...
{'Avg. MC Price Rho: 0','Avg. MC Price Rho: -0.3',...
'Avg. MC Price Rho: -0.7'});

V = permute(bls_final,[1 3 2]);
V = reshape(V,[],size(bls_final,2),1);
Avg_ImpVol_Table= array2table(V,'VariableNames', ...
{'Avg. MC ImpVol Rho: 0','Avg. MC ImpVol Rho: -0.3',...
'Avg. MC ImpVol Rho: -0.7'});

call_se_final = std(call,1);
bls_se_final = std(bls,1);

C_std = permute(call_se_final,[1 3 2]);
C_std = reshape(C_std,[],size(call_se_final,2),1);
Std_Prices_Table = array2table(C_std,'VariableNames', ...
{'std.err.MC Price Rho: 0','std.err.MC Price Rho: -0.3',...
'std.err.MC Price Rho: -0.7'});

V_std = permute(bls_se_final ,[1 3 2]);
V_std = reshape(V_std,[],size(bls_se_final ,2),1);
Std_ImpVol_Table= array2table(V_std,'VariableNames', ...
{'std.err.MC ImpVol Rho: 0','std.err.MC ImpVol Rho: -0.3',...
'std.err.MC ImpVol Rho: -0.7'});
```

2. **Estimate the stochastic volatility model's call price using conditional Monte Carlo and the formula of Romano and Touzi:**

$$C^{stoch-vol}(s, x, T) = E[C^{bs}(se^Z, \sigma(X), T) \mid S_0 = s, X_0 = x],$$

where $C^{bs}(s, \sigma, T)$ **denotes the Black-Scholes call price and**

$$Z = \rho \int_0^T \sqrt{X_t} dB_t - \frac{\rho^2}{2} \int_0^T X_t dt$$

$$\sigma^2(X) = \frac{1-\rho^2}{T} \int_0^T X_t dt.$$

Use Z **and** $\sigma^2(X)$ **as the conditional variables and use to estimate the call price. Generate 100 of these Conditional MC estimators and save in a** $7 * 100$ **array. Also save each estimator's implied volatility in a** $7 * 100$ **array.**

Solution:

```
clear
clc

X0 = 0.20;
S0 = 100;
kappa = 3;
theta = 0.2;
sigma = sqrt(2*theta*kappa);
T = 3/12;
r = 0.05;
rhos = [0, -0.3, -0.7];
K_values = 90:5:120;
plot_num = 1;
M = 20000;
N = round(T*365);
dt = T/N;
n = round(365*T);
steps = floor(T/dt);
L = 100;

call_price=zeros(M, length(rhos), length(K_values));
bls_vol=zeros(M, length(rhos), length(K_values));

call = zeros(L, length(rhos), length(K_values));
bls = zeros(L, length(rhos), length(K_values));
Z = ones(M, length(rhos));
S = ones(M, length(rhos));
sig = ones(M, length(rhos));
```

```
for i=1:L
    for k=1:length(K_values)
        for j=1:length(rhos)
            rho = rhos(j);
            X = X0*ones(M, steps);
            X(:, 1) = X0;
            %S = log(S0)*ones(M, steps);
            for t = 2:steps
                dW = randn(M, 1) * sqrt(dt);
                dB = randn(M, 1) * sqrt(dt);
                X(:, t) = (1 - kappa*dt) * X(:, t-1) + kappa*theta*dt ...
                + sigma .* sqrt(max(X(:, t-1), 0)) .* dW;
            end
            Z(:, j) = (rho/sigma)*((X(:,steps)-X0) - kappa*theta*T + ...
            kappa*sum(X(),2)*dt)-(rho^2/2)*sum(X(),2)*(dt);
            S(:,j) = S0 * exp(Z(:, j));
            sig(:,j) = sqrt(((1-rho^2)/T)*(sum(X(),2)*dt));
        end
        call_price(:,:,k) = blsprice(S, K_values(k), r, T, sig);
        %bls_vol(:,:,k) = blsimpv(S0, K_values(k), r, T, call_price(:,:,k));
    end
    if i==1
        call_temp = mean(call_price,1);
    end
    call(i,:,:) = mean(call_price,1);
    for k=1:length(K_values)
        bls(i,:,k) = blsimpv(S0, K_values(k), r, T, call(i,:,k));
    end
    %bls(i,:,:) = mean(bls_vol,1);
end

call_final = mean(call,1);
bls_final = mean(bls,1);

C = permute(call_final,[1 3 2]);
C = reshape(C,[],size(call_final,2),1);
Avg_Prices_Table = array2table(C,'VariableNames', {'Avg-C-MC Price ...
Rho: 0','Avg-C-MC Price Rho: -0.3','Avg-C-MC Price Rho: -0.7'});

V = permute(bls_final,[1 3 2]);
V = reshape(V,[],size(bls_final,2),1);
Avg_ImpVol_Table= array2table(V,'VariableNames', {'Avg-C-MC ImpVol ...
Rho: 0','Avg-C-MC ImpVol Rho: -0.3','Avg-C-MC ImpVol Rho: -0.7'});
```

```
call_se_final = std(call,1);
bls_se_final = std(bls,1);

C_std = permute(call_se_final,[1 3 2]);
C_std = reshape(C_std,[],size(call_se_final,2),1);
Std_Prices_Table = array2table(C_std,'VariableNames', {'std.err.C-MC ...
Price Rho: 0','std.err.C-MC Price Rho: -0.3',...
'std.err.C-MC Price Rho: -0.7'});

V_std = permute(bls_se_final ,[1 3 2]);
V_std = reshape(V_std,[],size(bls_se_final ,2),1);
Std_ImpVol_Table= array2table(V_std,'VariableNames', {'std.err.C-MC ...
ImpVol Rho: 0','std.err.C-MC ImpVol Rho: -0.3',...
'std.err.C-MC ImpVol Rho: -0.7'});
```

3. **With the samples collected in parts 1. and 2., fill in Tables 2 and 3, and comment on any reduction in variance that is seen in the tables. Use Matlab's optByHeston to obtain the Heston price.**

   Solution:

```
clear
clc

X0 = 0.20;
S0 = 100;
kappa = 3;
theta = 0.2;
sigma = sqrt(2*theta*kappa);
T = 3/12;
r = 0.05;
rhos = [0, -0.3, -0.7];
K_values = 90:5:120;

N = round(T*365);
dt = T/N;
n = round(365*T);
steps = floor(T/dt);

call_price=zeros(length(rhos), length(K_values));
bls_vol=zeros(length(rhos), length(K_values));

Settle = datetime(2017,8,1);
Maturity = datetime(2017,10,1);

for k=1:length(K_values)
    for j=1:length(rhos)
        rho = rhos(j);
        call_price(j,k) = optByHestonNI(r, S0, Settle, Maturity, ...
        'call', K_values(k), X0, theta, kappa, sigma, rho);
        bls_vol(j,k) = blsimpv(S0, K_values(k), r, T, call_price(j,k));
    end
end

Heston_Prices_Table = array2table(call_price','VariableNames', ...
{'Avg-Heston Price Rho: 0','Avg-Heston Price Rho: -0.3',...
'Avg-Heston Price Rho: -0.7'});
Heston_StdVol_Table = array2table(bls_vol','VariableNames', ...
{'Std-Heston IV Rho: 0','Std-Heston IV Rho: -0.3',...
'Std-Heston IV Rho: -0.7'});
```

Output:

As rho is increasing either positive or negative direction the sigma term in the Romano and Touzi equation, will decrease because of the squared rho term. Hence there is more variance reduction when $\rho = 0$ and less when $|\rho|$ increases.

Analysis of Prices Rho=0					
Strike	Heston Price	Avg. MC	Avg. C-MC	std. err. MC	std. err. C-MC
90	13.4613	14.8574	14.9279	0.1314	0.0151
95	10.1723	11.7283	11.7995	0.1202	0.0158
100	7.4705	9.0991	9.1712	0.1097	0.0157
105	5.3561	6.9615	7.0322	0.0985	0.0176
110	3.7707	5.2736	5.3422	0.0878	0.0161
115	2.6217	3.9707	4.0313	0.0777	0.0160
120	1.8092	2.9804	3.0342	0.0685	0.0111

Figure 3: The CMC prices in this table have the least variance reduction. The ratios of MC to CMC standard errors are (8.678, 7.630, 6.970, 5.584, 5.464, 4.847, 6.148).

Analysis of Prices Rho=-0.3					
Strike	Heston Price	Avg. MC	Avg. C-MC	std. err. MC	std. err. C-MC
90	13.6381	15.0579	15.1220	0.1227	0.0227
95	10.2722	11.8332	11.9029	0.1130	0.0166
100	7.4513	9.0696	9.1382	0.1020	0.0118
105	5.2073	6.7880	6.8527	0.0902	0.0098
110	3.5173	4.9727	5.0314	0.0801	0.0079
115	2.3094	3.5802	3.6324	0.0697	0.0064
120	1.4842	2.5450	2.5899	0.0604	0.0060

Figure 4: The CMC prices in this table have the least variance reduction. The ratios of MC to CMC standard errors are (5.395, 6.823, 8.651,9.160, 10.113, 10.844, 10.040).

Analysis of Prices Rho=-0.7

Strike	Heston Price	Avg. MC	Avg. C-MC	std. err. MC	std. err. C-MC
90	13.8464	15.3083	15.3422	0.0967	0.0657
95	10.3933	11.9688	12.0181	0.0865	0.0530
100	7.4259	9.0367	9.0961	0.0762	0.0461
105	5.0004	6.5513	6.5994	0.0655	0.0299
110	3.1409	4.5352	4.5728	0.0543	0.0250
115	1.8255	2.9852	3.0163	0.0434	0.0155
120	0.9794	1.8657	1.8918	0.0338	0.0113

Figure 5: The CMC prices in this table have the least variance reduction. The ratios of MC to CMC standard errors are (1.471, 1.632, 1.652, 2.188, 2.170, 2.804, 2.980).

Analysis of Implied Volatilities Rho=0

Strike	Heston Price	Avg. MC	Avg. C-MC	std. err. MC	std. err. C-MC
90	0.3422	0.4321	0.4364	0.0082	0.0009
95	0.3435	0.4290	0.4330	0.0065	0.0009
100	0.3450	0.4278	0.4313	0.0056	0.0008
105	0.3475	0.4283	0.4319	0.0049	0.0009
110	0.3509	0.4304	0.4339	0.0046	0.0008
115	0.3551	0.4338	0.4373	0.0044	0.0009
120	0.3598	0.4379	0.4414	0.0043	0.0007

Figure 6: The CMC prices in this table have the least variance reduction. The ratios of MC to CMC standard errors are (8.678, 7.630, 6.970, 5.584, 5.464, 4.847, 6.148).

Analysis of Implied Volatilities Rho=-0.3

Strike	Heston Price	Avg. MC	Avg. C-MC	std. err. MC	std. err. C-MC
90	0.3540	0.4445	0.4486	0.0076	0.0014
95	0.3490	0.4347	0.4384	0.0061	0.0009
100	0.3441	0.4263	0.4297	0.0052	0.0006
105	0.3400	0.4196	0.4229	0.0045	0.0005
110	0.3372	0.4147	0.4178	0.0042	0.0004
115	0.3358	0.4116	0.4147	0.0040	0.0004
120	0.3356	0.4101	0.4130	0.0039	0.0004

Figure 7: The CMC implied volatilities in this table have least variance reduction. The ratios of MC to CMC standard errors are (5.405, 6.827, 8.650, 9.161, 10.122, 10.870, 10.084).

Analysis of Implied Volatilities Rho=-0.7					
Strike	Heston Price	Avg. MC	Avg. C-MC	std. err. MC	std. err. C-MC
90	0.3678	0.4599	0.4629	0.0059	0.0040
95	0.3557	0.4420	0.4452	0.0047	0.0029
100	0.3428	0.4246	0.4277	0.0039	0.0023
105	0.3295	0.4077	0.4102	0.0033	0.0015
110	0.3165	0.3917	0.3940	0.0029	0.0013
115	0.3044	0.3770	0.3790	0.0026	0.0009
120	0.2938	0.3639	0.3656	0.0024	0.0008

Figure 8: The CMC implied volatilities in this table have least variance reduction. The ratios of MC to CMC standard errors are (1.472, 1.632, 1.652, 2.188, 2.171, 2.811, 2.992).

3 Problem (Quasi Monte Carlo)

We want to estimate

$$\theta = \mathbb{E}cos\big(\|X\|\big),$$

where $X \in \mathbb{R}^{100}$ normally distributed vector with

$$X_i = \beta Z_0 + \sqrt{1-\beta^2} Z_i$$

for $i = 1, 2, \ldots, 100,$ where $Z_0 \sim iid$ normal(0, 1) and $\beta \in \big(-1, 1\big)$. Let us estimate θ with estimation of the type

$$\widehat{\theta} = \frac{1}{1000} \sum_{l=1}^{1000} cos\big(\|X^l\|\big),$$

where $X^l \in R^{100}$ is a sample. Perform the following analysis for $\beta = 0$ and again for $\beta = 0.8:$

1. **(Standard MC) Use standard MC to estimate θ, that is, generate 1000 normally distributed $X^l \in R^d$, compute $Y^l = cos\big(\|X^l\|\big)$ and take their sample average. Generate 1000 of these MC estimators and save in an array.**

2. **(1-Dimensional MC) Reduce the estimation to the expectation of a non-central chi-square random variable,**

$$\theta = EE\Bigg[cos\Bigg(\sqrt{\big(1-\beta^2\big)\chi^2_{100}\big(Z_0\big)}\Bigg) \mid Z_0\Bigg],$$

   where $\chi^2_{100}\big(Z_0\big)$ is a non-central chi-square random variable with DoF 100 and non-centrality $100 * \frac{\beta^2 Z_0^2}{1-\beta^2}$. Generate 1000 of these 1-D MC estimators and save in an array.

3. **(RQMC) Repeat part 1 using Z_i taken from the Sobol set after Matousek scrambling. Generate 1000 of these RQMC estimators and save in an array. Also save the non-randomized QMC estimator.**

For Standard MC, 1-Dimensional MC, and RQMC, generate a separate histogram of the estimators with a fitted bell curve, and overlaid vertical lines of the non-randomized QMC estimator. From these sample distributions, report the standard deviation for each estimator. Which estimator(s) appear to be unbiased? Which estimator appears to have the least variance?

Solution:

```
clear
clc
%% Parameters
d = 100; % dimension of X
n = 1000; % number of samples
beta = 0; % beta
L = 100;

%% QMC
% Generate Sobol points
sbl = sobolset(d+1,'Skip',1e3,'Leap',1e2);
Z = net(sbl,n);
Z = norminv(Z);

% Generate X for the current sample
X = beta*Z(:,1) + sqrt(1-beta^2)*Z(:,2:end);

% Compute theta_QMC for the current sample
Y = sqrt(sum(X.^2,2));
theta_qmc = mean(cos(Y));

%% RQMC
% Generate Sobol points
theta_mc = zeros(1,L);
theta_mc_1d = zeros(1,L);
theta_rqmc = zeros(1,L);

for ctr = 1:L
    % Monte Carlo d dimensions
    Z = randn(n,d+1);
    X = beta*Z(:,1) + sqrt(1-beta^2)*Z(:,2:end);
    Y = sqrt(sum(X.^2,2));
    theta_mc(ctr) = mean(cos(Y));
    % Monte Carlo 1 dimension
    Z0 = randn(n,1);
    Y = zeros(n,1);
    for i = 1:n
        lambda = d*(beta*Z0(i)/sqrt(1-beta^2))^2;
        Y(i) = sqrt(ncx2rnd(d, lambda, 1, 1))*sqrt(1-beta^2);
    end
    theta_mc_1d(ctr) = mean(cos(Y));
    % RQMC
    p = scramble(sbl,'MatousekAffineOwen');
    p = net(p, n);
    Z = norminv(p);
    X = beta*Z(:,1) + sqrt(1-beta^2)*Z(:,2:end);
    Y = sqrt(sum(X.^2,2));
    theta_rqmc(ctr) = mean(cos(Y));
end
```

```
% Compute the mean of all estimators for both cases
fprintf('(E[theta_mc],sd(theta_mc) = ...
(%1.4f,%1.4f)\n',mean(theta_mc),std(theta_mc))
fprintf('(E[theta_mc_1d],sd(theta_mc_1d) = ...
(%1.4f,%1.4f)\n',mean(theta_mc_1d),std(theta_mc_1d))
fprintf('(E[theta_qmc],sd(theta_qmc) = ...
(%1.4f,%1.4f)\n',mean(theta_qmc),std(theta_qmc))
fprintf('(E[theta_rqmc],sd(theta_rqmc) = ...
(%1.4f,%1.4f)\n',mean(theta_rqmc),std(theta_rqmc))

histfit(theta_mc,50)
hxl = xline(theta_qmc' ,'-', {'QMC'});
hxl.FontSize = 15;
hxl.LineWidth = 5;
title("Standard Monte Carlo");
xlabel("Estimator")
ylabel("Frequency")

histfit(theta_mc_1d,50)
hxl = xline(theta_qmc' ,'-', {'QMC'});
hxl.FontSize = 15;
hxl.LineWidth = 5;
title("1-dimensional MC");
xlabel("Estimator")
ylabel("Frequency")

histfit(theta_rqmc,50)
hxl = xline(theta_qmc' ,'-', {'QMC'});
hxl.FontSize = 15;
hxl.LineWidth = 5;
title("RQMC");
xlabel("Estimator")
ylabel("Frequency")
```

Output:

```
(E[theta_mc],sd(theta_mc) = (-0.6656,0.0107)
(E[theta_mc_1d],sd(theta_mc_1d) = (-0.6645,0.0128)
(E[theta_qmc],sd(theta_qmc) = (-0.6665,0.0000)
(E[theta_rqmc],sd(theta_rqmc) = (-0.6644,0.0126)
```

Figure 9: $\beta = 0$

```
(E[theta_mc],sd(theta_mc) = (0.2523,0.0228)
(E[theta_mc_1d],sd(theta_mc_1d) = (0.2507,0.0253)
(E[theta_qmc],sd(theta_qmc) = (0.2377,0.0000)
(E[theta_rqmc],sd(theta_rqmc) = (0.2494,0.0093)
```

Figure 10: $\beta = 0.8$

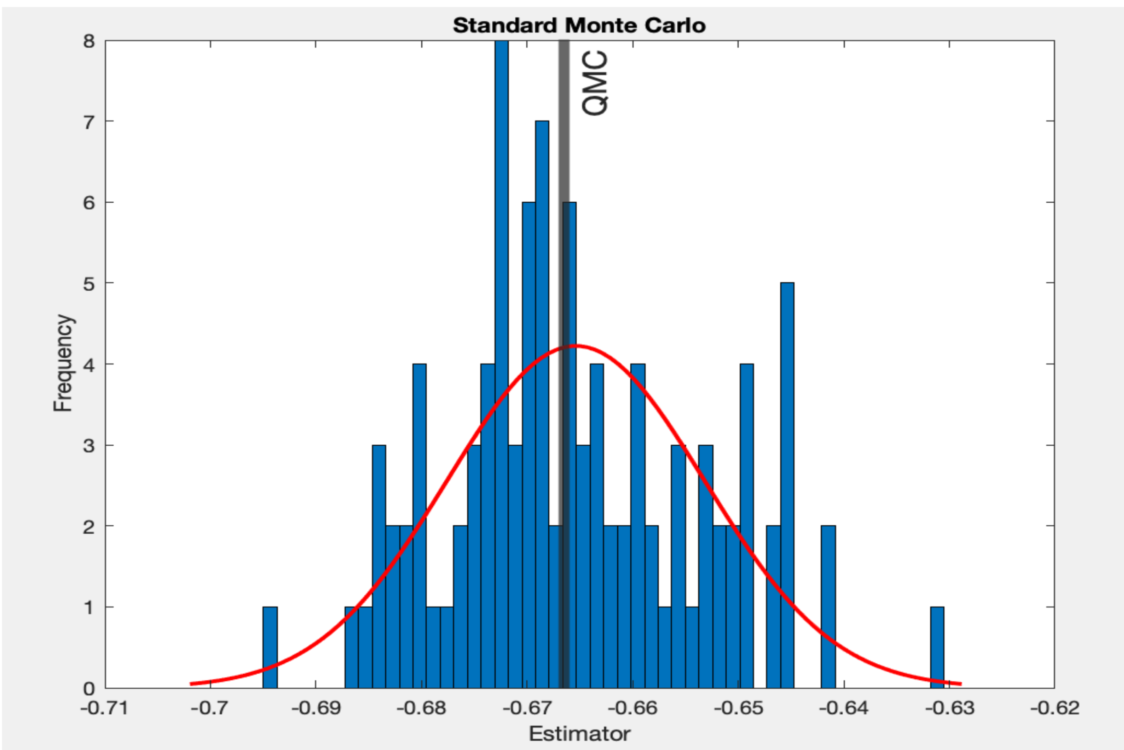

Figure 11: $\beta = 0$

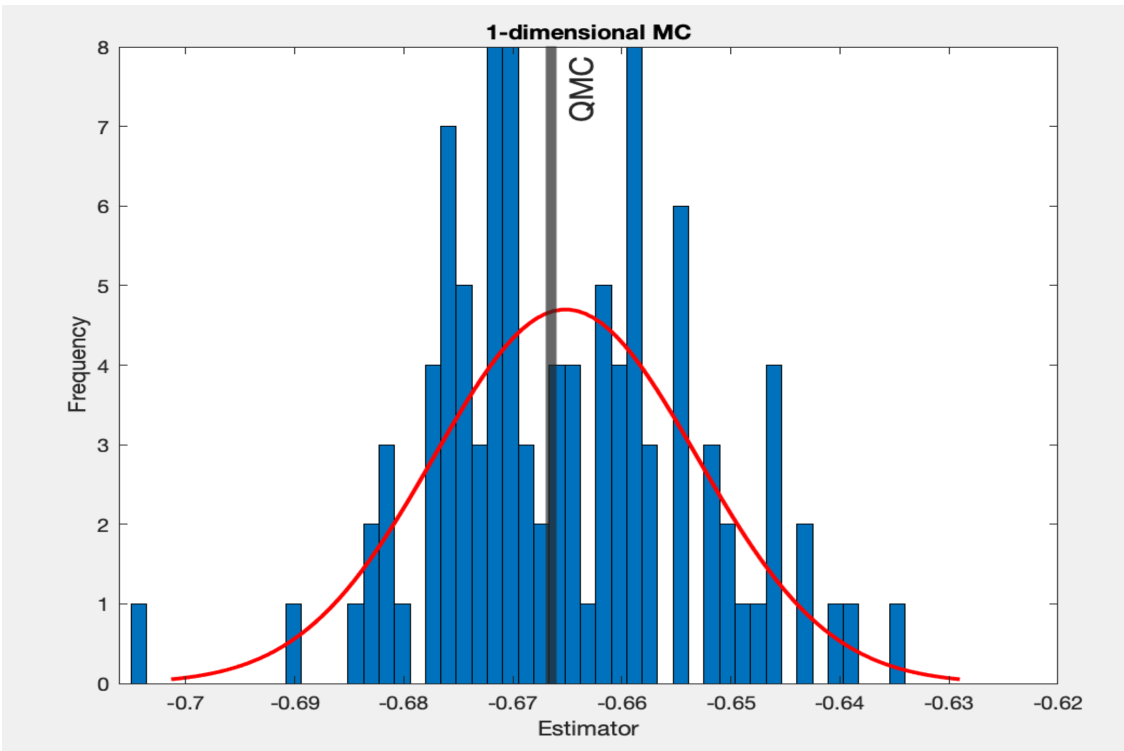

Figure 12: $\beta = 0$

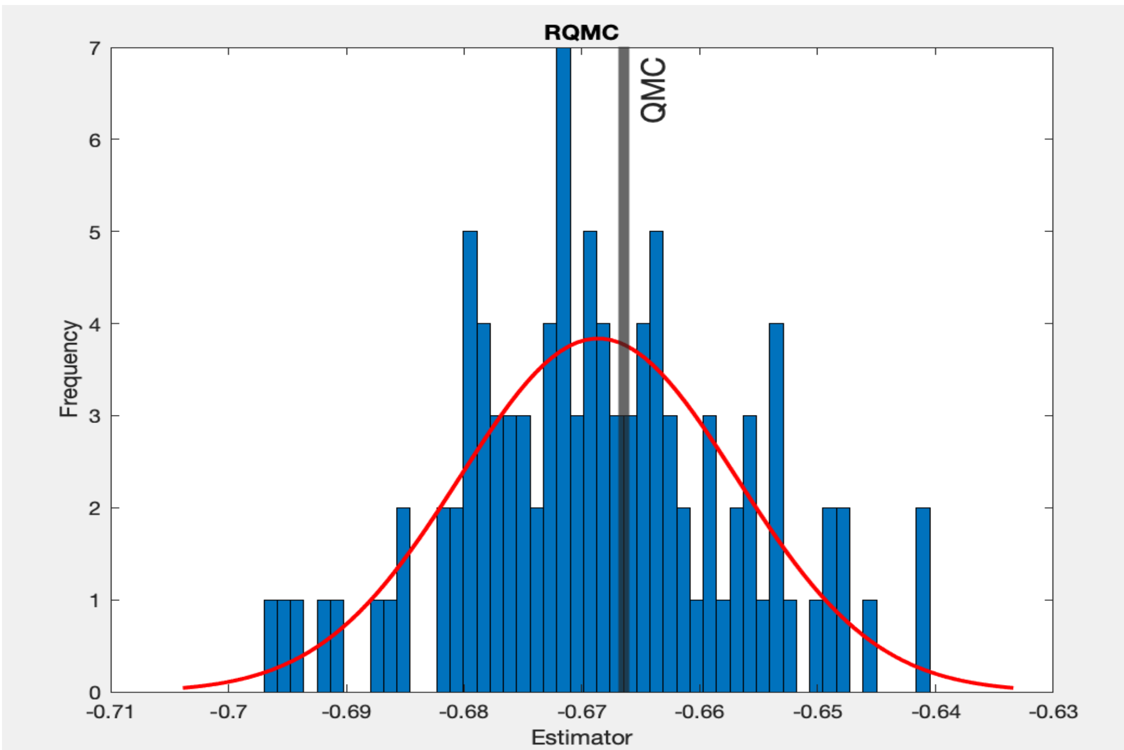

Figure 13: $\beta = 0$

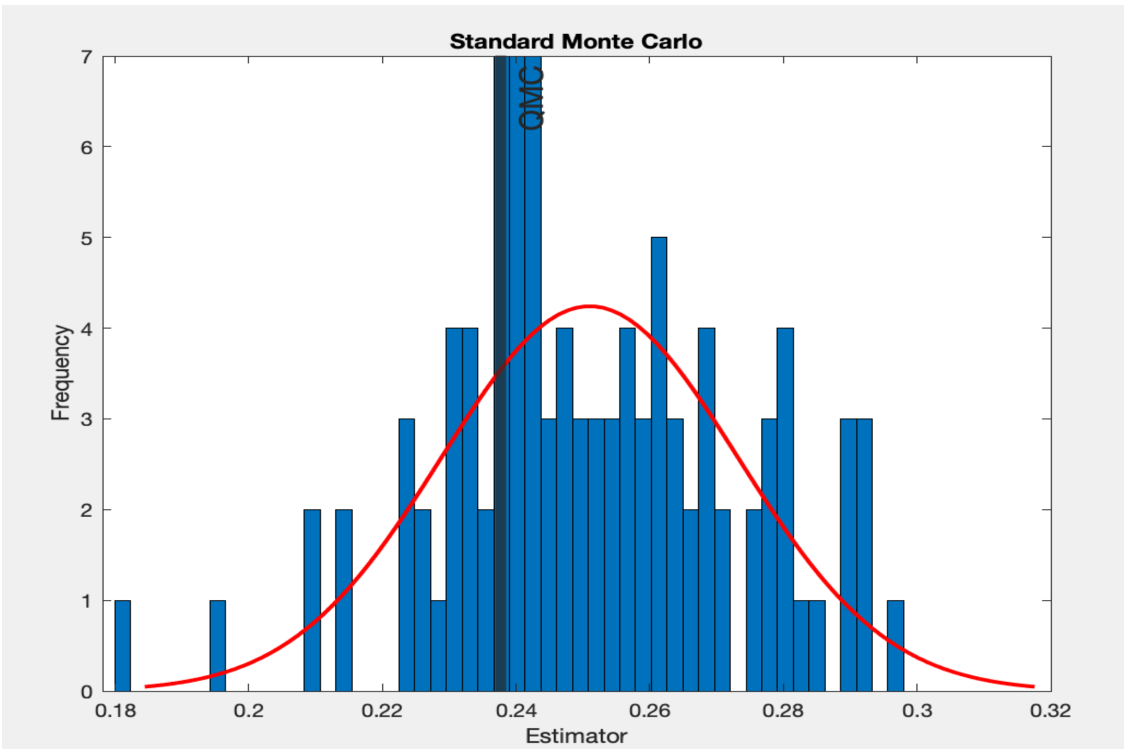

Figure 14: $\beta = 0.8$

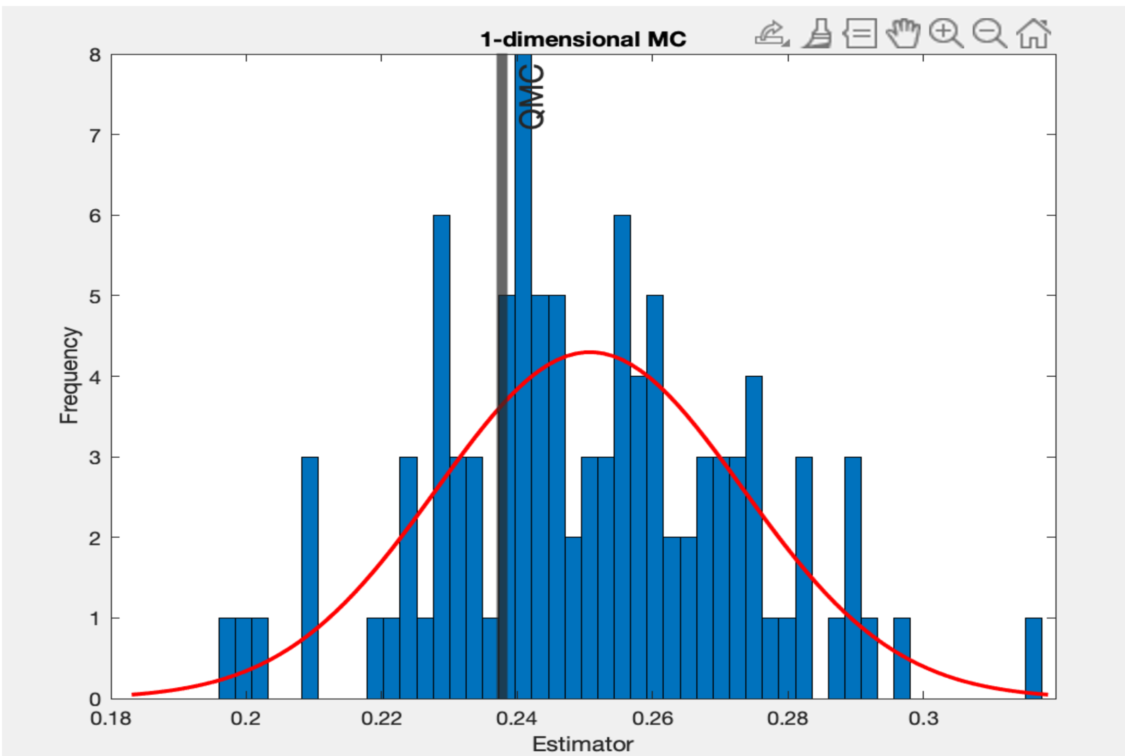

Figure 15: $\beta = 0.8$

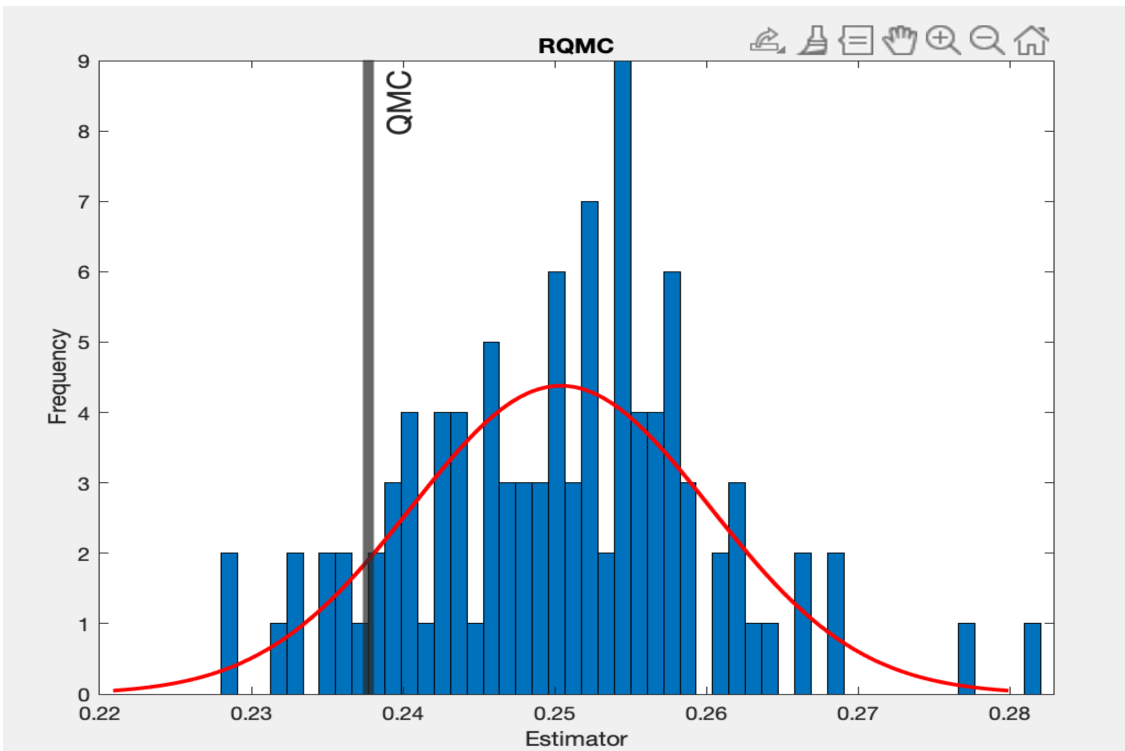

Figure 16: $\beta = 0.8$

Which estimator(s) appear to be unbiased?

In order for an estimator to be unbiased, its expected value must exactly equal the value of the population parameter. The bias of an estimator is the difference between the expected value of the estimator and the actual parameter value. Thus, if this difference is non-zero, then the estimator has bias. From all the graphs, we can see that the theta of Quasi Monte Carlo is approximately 1 standard deviation away from the mean value of the estimators in all the cases. Hence we are concluding from the Figures that all the estimators are unbiased.

Which estimator appears to have the least variance?

We can clearly see that Randomized Quasi Monte Carlo has the least variance because RQMC uses low-discrepancy sequences (LDS) instead of pseudo-random numbers. LDS are constructed to have more even coverage of the sample space compared to pseudo-random numbers, which tend to have clustering and gaps. This more even coverage can lead to better convergence properties and lower variance for certain types of integrals, including the one in this problem. RQMC can also be advantageous over standard MC when the integrand has a low effective dimension, which means that most of the variation in the integrand is concentrated in a lower-dimensional subspace. In this case, RQMC can more effectively sample the important regions of the sample space and lead to faster convergence than standard MC.