STA3431-Monte_Carlo_Methods-Homework_1

Name: Sergio E. Betancourt
Student Number: 998548585
Department: Statistical Sciences
Program: M.Sc. 2019
E-mail: sergio.betancourt@mail.utoronto.ca

Date: 2018-10-15

Question 1

a) Write a computer program to generate pseudorandom Uniform[0,1] numbers using a method of your choice. Your program should just use simple arithmetic, and should not use any built-in randomness functions. Explain your reasons for your choice of method.

Solution:

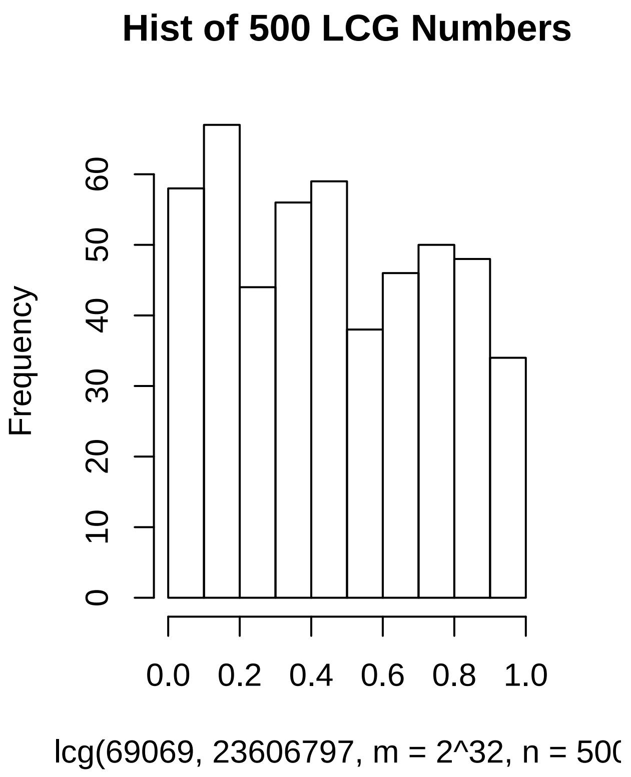
I consider the Linear Congruential Generator (LCG) that we discussed in class because it is easily implemented with simple arithmetic. Moreover, it performs well subject to the appropriate parameters, as it can have a large period and large cycles to support non-repetition, and it is quite fast. I am using system time in milliseconds as the seed value for every run.

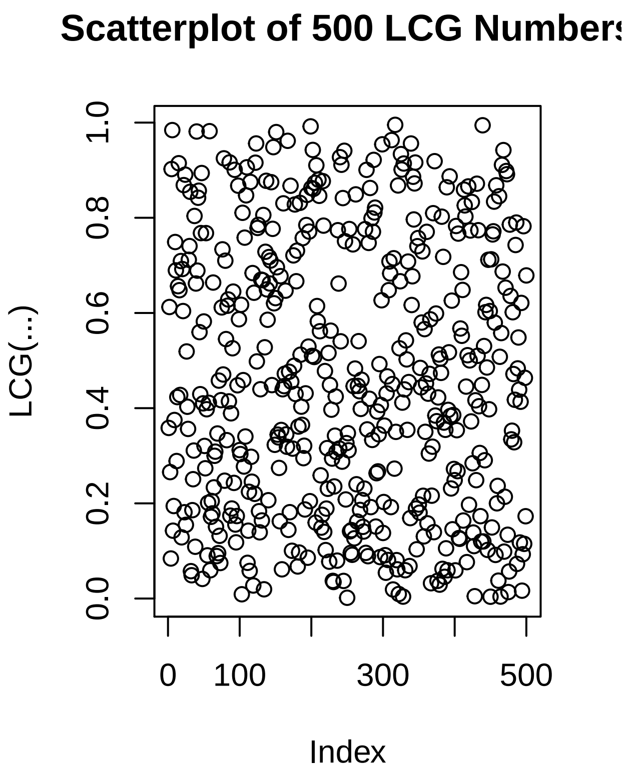
```r
lcg <- function(a,b,m,n){
  #a,b,m integers > 0
  #n number of pseudorandom values to return (n > 1)
  #Specify initialization value
  X <- round(as.numeric(Sys.time())); #initialization value (current time in milliseconds)
  seq0 <- X; sequ <- X/m;
  for (i in 2:n){
    X <- (a * seq0[i-1] + b) %% m
    seq0 = c(seq0,X)
    sequ = c(sequ,X / m)
  }
  #cat("Seed value for this run:",X,"\n");
  return(sequ);
};

lcg(69069,23606797,m=2^32,n=30);
```

```
##  [1] 0.35846193 0.61253616 0.26566301 0.08413177 0.90240164 0.98427858
##  [7] 0.14254136 0.19437620 0.37517184 0.74910139 0.68945958 0.28890555
## [13] 0.42325285 0.65635682 0.91467435 0.64792341 0.42777083 0.70916219
## [19] 0.12856255 0.69233084 0.60404614 0.86868459 0.18135321 0.89064775
## [25] 0.15485259 0.51916775 0.40300787 0.35636402 0.71176266 0.74057608
```

b) Use your program to generate and plot 500 independent pseudorandom Uni- form[0,1] numbers.

```r
par(mfrow=c(1,2));
hist(lcg(69069,23606797,m=2^32,n=500),main="Hist of 500 LCG Numbers")
plot(lcg(69069,23606797,m=2^32,n=500),main="Scatterplot of 500 LCG Numbers",ylab="LCG(...)");
```





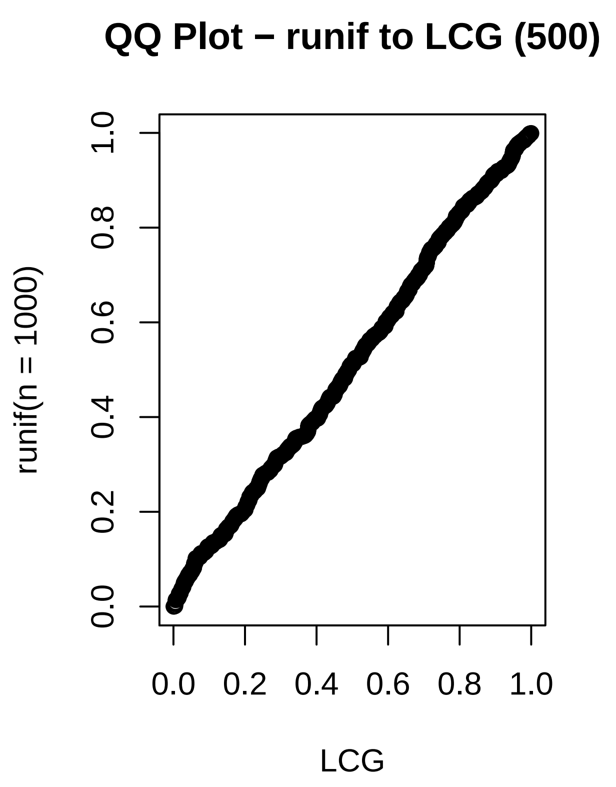
c) Perform (with explanation) a few statistical tests of your choosing to see how random/uniform/independent your generator "seems" to be.

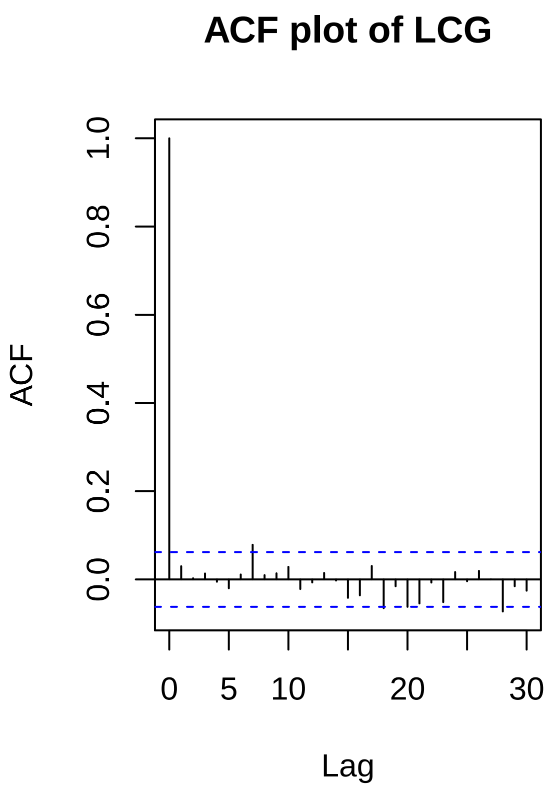
Solution:

To test how close to a uniform distribution my generator is, I can harness the qualitative usefulness of the QQ-plot. Using this plot I can compare the how close is the output of my generator to R's implementation of the uniform distribution. In this plot, the close resemblance to a straight line is evidence of the closeness of my pseudo-random numbers to R's implementation of a uniform[0,1].

In terms of independence, by examining the ACF graph, I can get an idea of the presence or lack of autocorrelation between numbers generated using this LCG, given the sequential nature of this algorithm.

```
vec <- lcg(69069,23606797,m=2^32,n=1000);par(mfrow=c(1,2));
qqplot(vec,runif(n=1000),main="QQ Plot - runif to LCG (500)",xlab="LCG");acf(vec,main="ACF plot of LCG")
```





To further test similarities in distribution between my pseudorandom generator and R's implementation of the uniform, I performed the below Kolmogorov-Smirnoff test–its high p-value means a lack of evidence to reject the null hypothesis that the distributions are different.

```
ks.test(lcg(69069,23606797,m=2^32,n=500),"punif")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  lcg(69069, 23606797, m = 2^32, n = 500)
## D = 0.025367, p-value = 0.9045
## alternative hypothesis: two-sided
```

Question 2

(a) Write a computer program to compute a good "classical" (i.i.d.) Monte Carlo estimate (including standard error and 95% confidence interval) of $E[YZ^4 \sin(YZ^2)]$, where Y ~ Exponential(3) and Z ~ Normal(0; 1) are independent. Your program should use your own pseudorandom function from the previous question, and should not use any built-in randomness functions.

Solution

I can get values from normal and exponential distributions by applying appropriate transformations to the above pseudo-random generator of values uniform[0,1].

By the method of inverse CDF I obtain the below scaled exponential from a Unif[0,1]:

$$U \sim unif[0,1] \Rightarrow -(\frac{1}{\lambda}) \log(U) \sim exp(\lambda), \ \lambda > 0$$

By the Box-Muller transform I obtain the below N(0;1) random variable from 2 independent uniformly distributed random variables:

$$U, V \sim unif[0,1] \Rightarrow Z = \sqrt{-2\log(U)}\ \cos(2\pi V) \sim N(0,1)$$

Then I can harness classical Monte Carlo to approximate the below expectation for $g(Y,Z) = YZ^4 \sin(YZ^2)$:

$$E[g(Y,Z)] \approx \frac{1}{M}\sum_{i=1}^{M} g(y_i, z_i)$$

```
#Getting n exp(3) samples from n unif[0,1]
exponentiate = function(xxx,rate) {
  return( -(1/rate)*log(xxx))  # positive Exponential r.v. with rate r
}

#Getting X ~ N(0;1) from 2 independent U,V ~ unif[0,1]
normalize = function(nsample){
  input1 <- lcg(69069,23606797,m=2^32,n=nsample);
  input2 <- lcg(69069,23606797,m=2^32,n=nsample);

  output = rep(NA,nsample)
  #Produce two independent, standard normal random variables from 2 independent unif[0,1], but discard
  for (i in 1:nsample) {
    us = c(input1[i],input2[i])
    R = sqrt(-2*log(us[1]))
    theta = 2*pi*us[2]
    output[i] = R*cos(theta)
    #samples[2*sim-1] = R*sin(theta)
  }
  return(output)
}

#Now, compute Monte Carlo estimate (nsim=80000)
M <- 80000
basic <-  lcg(69069,23606797,m=2^32,n=M);
Y <- exponentiate(basic,3);
Z <- normalize(M);

g <- function(Y,Z){
  return(Y*Z^4*sin(Y*Z^2))
}

funcc <- g(Y,Z)

cat("Monte Carlo estimate: ", mean(funcc),"\n");
```

```
## Monte Carlo estimate:  0.3293494
```

```
cat("Std error of the MC estimate:", sd(funcc)/sqrt(M),"\n");
```

```
## Std error of the MC estimate: 0.01240943
```

```
cat("95% CI for the above estimate:", c(mean(funcc)-qnorm(.975)*(sd(funcc)/sqrt(M)),mean(funcc)+qnorm(.9
```

```
## 95% CI for the above estimate: 0.3050274 0.3536715
```

(b) Run your program several times, and produce a final estimate.

Solution

```
M <- 80000;
reps <- 10;
finalvec <- numeric(reps);
sevec <- numeric(10);

g <- function(Y,Z){
  return(Y*Z^4*sin(Y*Z^2)) #original function times reciprocal of normal and exp(3)
}

for (i in 1:reps){
  storage <- NULL
  basic <-  lcg(69069,23606797,m=2^32,n=M);
  Y <- exponentiate(basic,3)
  Z <- normalize(M)
  storage <- g(Y,Z)
  finalvec[i] <- mean(storage)
  sevec[i] <- sd(storage)/sqrt(M)
};

cat("Vector of 10 MC estimates of M=80000 each:", finalvec,"\n");
```

```
## Vector of 10 MC estimates of M=80000 each: 0.3304251 0.3126939 0.3128164 0.3097238 0.3455092 0.31490
```

```
cat("Final MC estimate (mean of above 10 runs):", mean(finalvec),"\n");
```

```
## Final MC estimate (mean of above 10 runs): 0.325128
```

(c) Discuss how accurate you think your estimate is.

Solution:

Examining the standard errors below from the 10 runs used in part b) to produce a final estimate, we notice low variability from run to run. The standard errors could be tightened further with larger M's but my equipment is already struggling with the above 10 runs of 2M 80000 sample each. For reference, the value that http://wolframalpha.com yields after numerical integration is **0.320385**. Below is expression to evaluate the integral numerically: `\int_0^\infty \left( \int_{-\infty}^{\infty} y*(z^4)*sin(y*z^2) * 1/sqrt(2*pi) * exp((-z^2)/2) * (3)exp(-3*y) dz \right) dy`

```
cat("These are the standard errors of the above 10 runs of the MC algorithm:",sevec,"\n")
```

```
## These are the standard errors of the above 10 runs of the MC algorithm: 0.01158644 0.01126179 0.0118
```

Question 3

Re-write the given integral ("I") as some expected value and estimate it using a Monte Carlo algorithm of your choice. Discuss how well the algorithm works, produce a final estimate, and discuss how accurate do you believe such an estimate to be.

Solution:

Given the domains of integration in the double integral, I re-write "I" as an expectation of the integrand function, evaluated in terms of a standard normal random variable (with support $(-\infty, \infty)$) and a shifted exponential random variable (with support $(1, \infty)$).

We have:

$$f(X,Y) = (1 + X^3 + (Y-3)^2 + sin(XY^2))^{-|Y|^3-2}$$
$$g(X) = e^{-(X-1)}$$
$$r(Y) = \frac{1}{\sqrt{2\pi}} e^{-Y^2/2}$$

Then I can approximate the following using classical Monte Carlo: $I = E(f(X,Y)\, g(X)^{-1}\, r(Y)^{-1})$

To compute the above, I write my own function to sample from the shifted exponential distribution, given how R does not have it in its core functions. Such a method allows me to generate values from a unif[0,1] using R's `runif()` function, and use them to sample from a desired distribution. To sample from the standard normal I use R's `rnorm()`.

```r
# Support of exponential is (0,Inf), but want a shifted exponential with support (1,Inf). Use inverse C

K<-300000;

exponentiate_v2 = function(xxx,rate,shift) {
  return( shift - (1/rate)*log(xxx))
};

basic2 <- runif(K,0,1);

#For the second variable we can simulate it using R's own rnorm():

Y <- rnorm(K);
X <- exponentiate_v2(basic2,1,1);

#Use shifted exponential and std normal to compute integral as expectation. Employ classical Monte Carl

integrand_f <- function(x,y){
  return( (sqrt(2*pi)*exp((y^2)/2) ) *
          ( exp(x-1) )
          *(1 + x^3 + (y-3)^2 + sin(x*(y^2)))^(-1*(abs(y)^3)-2) )
};

q3builder <- integrand_f(X,Y);

cat("Arbitrary Monte Carlo estimate: ", mean(q3builder),"\n");
```

```
## Arbitrary Monte Carlo estimate:  0.01069216
```

```r
cat("Std error of the above MC estimate:", sd(q3builder)/sqrt(K),"\n");
```

```
## Std error of the above MC estimate: 4.321729e-05
```

```r
cat("95% CI for the above estimate:", c(mean(q3builder)-qnorm(.975)*sd(q3builder)/sqrt(K),mean(q3builde
```

```
## 95% CI for the above estimate: 0.01060746 0.01077687
```

To evaluate my method I compare it to the numerical integration I carried out using http://wolframalpha.com. The below expression yields **0.0106244**, which my Monte Carlo estimate approximates extremely well:

```
\int_1^\infty \left( \int_{-\infty}^{\infty} (1+x^3 + (y-3)^2 + sin(xy^2))^{-|y|^3 -2}dy
\right) dx
```

Question 4

(a) Let A=8, B=5, C=8, D=5 given that my student number is 998548585.

(b) Use an importance sampler to estimate $E_\pi[\frac{(X_1-X_2)}{(2+X_3+X_4+X_5)}]$.

Solution:

Let $h(X_1, X_2, X_3, X_4, X_5) = \frac{(X_1-X_2)}{(2+X_3+X_4+X_5)}$, and consider $g(x_1, x_2, x_3, x_4, x_5)$ and $\pi(x_1, x_2, x_3, x_4, x_5) = c\, g(x_1, x_2, x_3, x_4, x_5)$ as given in the problem statement.

Unfortunately I was not able to compute the target expectation numerically, so I shall compare the algorithms in part b) and c) in terms of perceived run-time and varibility.

Importance sampling requires that I find a suitable f. Here I consider $f(X_1, X_2, X_3, X_4, X_5) = \prod_{i=1}^{5} \mathbf{1}_{0<x_i<1}$. Not only does this f have the desired support but sampling from it is straightforward (either using my LCG or R's unif[0,1] generator). Moreover, as discussed during lecture in regards to importance sampling, I shall use the same samples for the numerator and denominator in order to control variability and improve computability.

Then I can sample from f and calculate my approximation to "I" by calculating the ratio of numerator and denominator, as below:

$$I \approx \frac{\sum_{i=1}^{M}(h(x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i}, x_{5,i})\, g(x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i}, x_{5,i}))}{\sum_{i=1}^{M} g(x_{1,i}, x_{2,i}, x_{3,i}, x_{4,i}, x_{5,i})}$$

```r
K=100000;
sim1=30;
sim2=5;
set.seed(12345);

A=8; B=5; C=8; D=5;

g10 <- function(x1,x2,x3,x4,x5){
  return (((x1+A+2)^(x2+3)) * (1+cos(2*x2 + 3*x3 + 4*x4 + (B+3)*x5)) *
    exp((12-C)*x4) * (abs(x4 - 3*x5)^(D+2)))
    }

r10 <- function(x){
  return (exp((12-C)*x[1]) * (abs(x[1] - 3*x[2])^(D+2)))
}

h10 <- function(x1,x2,x3,x4,x5){
  return ((x1 - x2) / (2 + x3 + x4 * x5))
};

result <- numeric(sim1);
f_result <- numeric(sim2);
f_sd <- numeric(sim2);

for (j in 1:sim2){
  for (i in 1:sim1){
    x1 <- runif(K); x2 <- runif(K); x3 <- runif(K); x4 <- runif(K); x5 <- runif(K);
    result[i] <- mean(h10(x1,x2,x3,x4,x5)*g10(x1,x2,x3,x4,x5))/mean(g10(x1,x2,x3,x4,x5))
  }
  cat("Run",j,"out of",sim2,"\n")
  cat("Monte Carlo estimate: ", mean(result),"\n");
  cat("Std error of the above MC estimate:", sd(result)/sqrt(sim1),"\n");
  cat("95% CI for the above estimate:", c(mean(result)-qnorm(.975)*sd(result)/sqrt(sim1),mean(result)+q
  f_result[j] <- mean(result);
  f_sd[j] <- sd(result)/sqrt(sim1);

};
```

```
## Run 1 out of 5
## Monte Carlo estimate:  -0.05322298
## Std error of the above MC estimate: 0.0001925204
## 95% CI for the above estimate: -0.05360032 -0.05284565
##
## Run 2 out of 5
## Monte Carlo estimate:  -0.0534588
## Std error of the above MC estimate: 0.0002471621
## 95% CI for the above estimate: -0.05394322 -0.05297437
##
## Run 3 out of 5
## Monte Carlo estimate:  -0.05301001
## Std error of the above MC estimate: 0.0002631504
## 95% CI for the above estimate: -0.05352577 -0.05249424
##
## Run 4 out of 5
## Monte Carlo estimate:  -0.05329292
## Std error of the above MC estimate: 0.0001968904
## 95% CI for the above estimate: -0.05367882 -0.05290702
##
## Run 5 out of 5
## Monte Carlo estimate:  -0.05331201
## Std error of the above MC estimate: 0.0002278445
## 95% CI for the above estimate: -0.05375858 -0.05286544
```

```
cat("Final Monte Carlo estimate:", mean(f_result),"\n");
```

```
## Final Monte Carlo estimate: -0.05325934
```

```
cat("Standard errors for the 5 above runs",f_sd,"\n")
```

```
## Standard errors for the 5 above runs 0.0001925204 0.0002471621 0.0002631504 0.0001968904 0.000227844
```

Importance sampling yields a substantially low error (compared to part c), and a comparatively fast run-time. Sampling 100000 observations 30 times, on 5 different simulations yielded certain variability in the standard errors, but not to the large extend of part c).

(c) The objective is to use a rejection sampler to estimate $E_\pi[(X1 - X2)/(2 + X3 + X4 + X5)]$.

Solution:

In order to find the appropriate K used in this algorithm, I bound the function g piecewise as below:

$$(x_1 + 10)^{x_2+3} \leq 11^4$$

$$1 + \cos(...) \leq 2$$

$$e^{4x_4}|x_4 - 3x_5|^7 \leq e^4 2^7$$

$$K = 2 \times 11^4 \times 2^7 e^4 = 204639108$$

Note that $g(X_1, X_2, X_3, X_4, X_5) \leq 204639108$ for my values of A, B, C, D, so let K equal this number. Moreover, we let $f(X_1, X_2, X_3, X_4, X_5) = \prod_{i=1}^{5} \mathbf{1}_{0<x_i<1}$, by the same reasons stated in part a), plus noting how $Kf \geq g(X_1, X_2, X_3, X_4, X_5)$

Let $h(X_1, X_2, X_3, X_4, X_5) = \frac{(X_1 - X_2)}{(2+X_3+X_4+X_5)}$.

```
set.seed(12345);
A=8; B=5; C=8; D=5;

M=80000;
K=204639108;
sim=5;
accrate <- numeric(sim);

g <- function(x1,x2,x3,x4,x5){
  return (((x1+A+2)^(x2+3)) * (1+cos(2*x2 + 3*x3 + 4*x4 + (B+3)*x5)) *
    exp((12-C)*x4) * (abs(x4 - 3*x5)^(D+2)))
    }

h <- function(x1,x2,x3,x4,x5){
  return ((x1 - x2) / (2 + x3 + x4 * x5))
};

for (j in 1:sim){
  hlist = alllist = NULL; xlist = NULL;
  samm <- numeric(sim);
  for (i in 1:M){
    U <- runif(1);
    x1 <- runif(1); x2 <- runif(1); x3 <- runif(1); x4 <- runif(1); x5 <- runif(1);
    X <- c(x1,x2,x3,x4,x5);
    acc <- g(x1,x2,x3,x4,x5) / (K);
    alllist <- c(alllist,X)
    if (U < acc){
      xlist <- rbind(xlist,X);
      hlist <- c(hlist,h(x1,x2,x3,x4,x5));
    }
  }
  accrate[j] <- length(xlist[,1])/M
  cat("Run",j,"out of",sim,"\n");
  cat("Out of", M,"rounds, produced", length(xlist[,1]), "5-dim samples\n")
  cat("Mean of h(X) is approximately",mean(hlist),"\n")
  cat("Standard error of h(X) is approximately",sd(hlist)/sqrt(length(hlist)),"\n\n")
  samm[j] <- mean(hlist);
}
```

```
## Run 1 out of 5
## Out of 80000 rounds, produced 994 5-dim samples
## Mean of h(X) is approximately -0.05297638
## Standard error of h(X) is approximately 0.003912998
##
## Run 2 out of 5
## Out of 80000 rounds, produced 990 5-dim samples
## Mean of h(X) is approximately -0.0589669
## Standard error of h(X) is approximately 0.003890739
##
```

```
## Run 3 out of 5
## Out of 80000 rounds, produced 1008 5-dim samples
## Mean of h(X) is approximately -0.05801856
## Standard error of h(X) is approximately 0.004075107
##
## Run 4 out of 5
## Out of 80000 rounds, produced 1044 5-dim samples
## Mean of h(X) is approximately -0.05679985
## Standard error of h(X) is approximately 0.0039115
##
## Run 5 out of 5
## Out of 80000 rounds, produced 1073 5-dim samples
## Mean of h(X) is approximately -0.06090873
## Standard error of h(X) is approximately 0.003853242
```

```r
cat("My mean acceptance rate in these 5 runs is",mean(accrate),"\n")
```

```
## My mean acceptance rate in these 5 runs is 0.0127725
```

I used the same seed (12345) for the above two parts of question 4 to compare their results fairly, as I am using R's built-in random generation. With such a large K, this technique yields a low acceptable sample size (around 1%), which contributes to the larger standard errors. To this end, **importance sampling** can yield more accurate estimates with comparable run-time.

A major limitation in this investigation is limited computing power, which curtails the use of higher Ks and Ms to obtain better approximations. However, procedures of this kind should also be measured on how quickly they can approach the answer with limited resources.