# Computational Statistics Coursework 1

Lorenz Wolf (CID 01351308)

June 16, 2021

## Question 1

In this question we will consider a random variable with cdf F, where F(x)=0 for $x < 0$, F(x)=1 for $x > 3$ and

$$F(x) = \frac{1 - e^{-2x}}{1 - e^{-6}},\ x \in [0, 3].$$

### a)

First we will use inversion to sample from a random variable with cdf F(.). To apply the algorithm we need to invert the distribution function to find that

$$F^{-1}(x) = \frac{-1}{2} ln(1 - (1 - e^{-6})x)\ x \in [0, 1].$$

For the inversion algorithm we first sample $U\ Uniform(0, 1)$ and then set $X = F^{-1}(U)$ to obtain a sample from the random variable X with distribution function F(.).

Having implemented the inversion method we will now compare the empirical distribution function to the theoretical distribution function for a sample of size 1000. For this we plot both, the empirical cdf and the theoretical cdf for a sample size of 1000 in Figure 1. We observe that the empirical cdf in black deviates quite significantly from the theoretical cdf between approximately x=0.2 and x=0.6. However, since we are using inversion we would expect this discrepancy to disappear as we increase the sample size further.

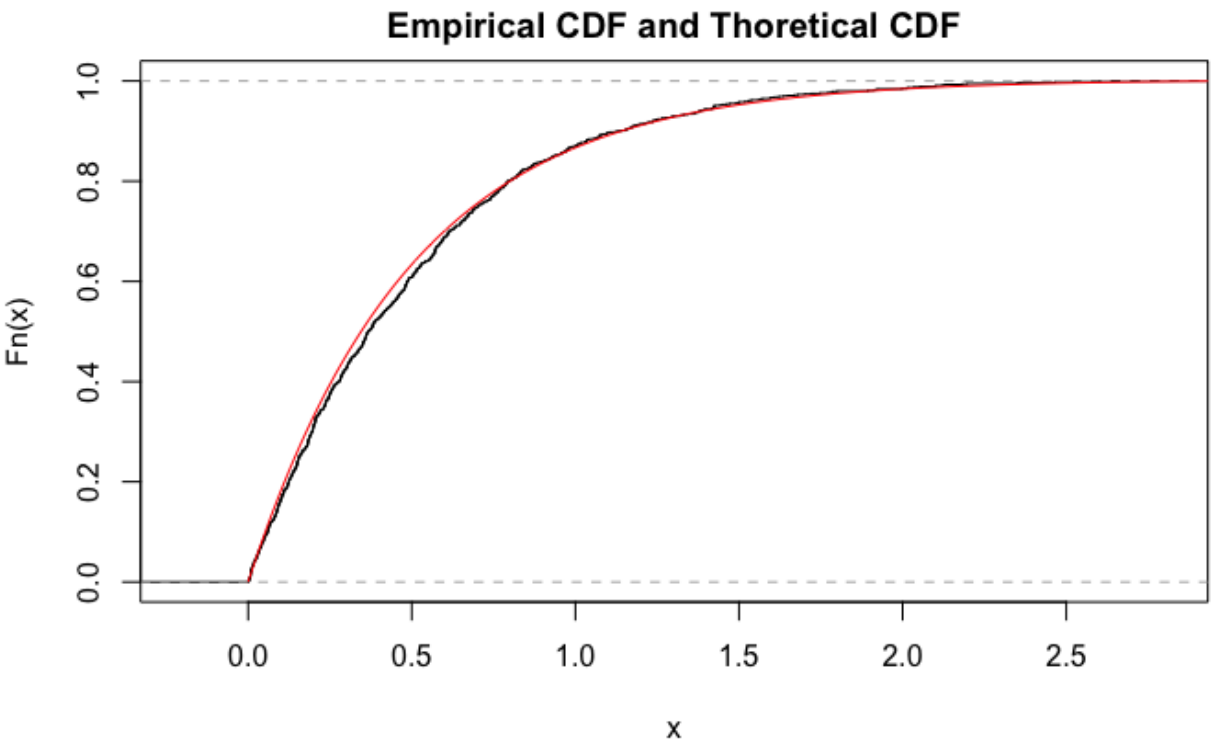


Figure 1: Empirical distribution function (black) and theoretical distribution function (red) a sample size of 1000.

### b)

We will now use this inversion algorithm to evaluate E(cos(X)) where X has cdf F(.) defined in part a) using Monte Carlo Integration. We will then check the estimate with numerical integration.

By Monte Carlo integration theory from lectures we have that the Monte Carlo estimator for E(cos(X)) is

$$mc_n = \frac{1}{n}\sum_{i=1}^{n} cos(X_i)$$

where $X_1, \ldots, X_n$ are idd distributed with cdf F(.).

We sample $X_i$ using the inversion algorithm from part a) and compute $mc_{10^4} \approx 0.8049$.

Now let us check this estimate with numerical integration. First differentiating F(x) to find the density function and then using numerical integration we compute

$$\int_0^3 cos(x)\frac{2e^{-2x}}{1-e^{-6}}dx \approx 0.8041.$$

This seems to be fairly close to the Monte Carlo estimate.

### c)

To find a 95% confidence interval we refer to theory from lectures. In particular we know that

$$[mc_n + \frac{1}{\sqrt{n}}Sz_{\alpha/2}, mc_n + \frac{1}{\sqrt{n}}Sz_{1-\alpha/2}]$$

where $S^2$ is the sample variance and $z_\alpha$ is such that $P(X < z_\alpha = \alpha$ with $X \sim N(0,1)$, is a 95% confidence interval for $mc_n$. We report the 95% confidence interval $[0.799, 0.811]$.

To obtain a confidence interval with length no larger than $10^{-6}$ we will need a sample size of approximately $10^{12.2}$. This follows from the fact that the length of the interval is linear in the standard error of the estimate which is $O(\frac{1}{\sqrt{n}})$. In particular we have that $l = 2\frac{S}{\sqrt{n}}z_{1-\alpha/2}$. We know that for $n = 10^4$ we obtain $l \approx 0.125$ and the result follows by simple algebra.

## Question 2

Let $X \sim Gamma(2,3)$. In this question we will use importance sampling to estimate $I_1 = P(X > 2)$ and $I_2 = E(\frac{1}{X})$. Let $f(x) = \frac{3^2}{\Gamma(2)}xe^{-3x}$ for $x \in (0,\infty)$ and 0 otherwise, furthermore let g(x) be the importance sampling distribution and let $Y_1, \ldots, Y_n \sim g$ be independently and identically distributed. We then have that the importance sampling estimators are

$$\hat{I}_{1,n} = \sum_{i=1}^{n} I(Y_i > 2)\frac{f(Y_i)}{g(Y_i)}$$

and

$$\hat{I}_{2,n} = \sum_{i=1}^{n} \frac{1}{Y_i}\frac{f(Y_i)}{g(Y_i)}.$$

### a)

First we will use a standard normal distribution as importance sampling distribution to estimate $I_1$ and $I_2$. Based on a sample size of $10^4$ we report the corresponding estimates, standard errors and 95% confidence intervals in Table 1.

| | n | Estimate | Standard Error | 95% Confidence Interval |
|---|---|---|---|---|
| $I_1$ | $10^4$ | 0.0194 | $1.201 \times 10^{-3}$ | [0.017, 0.0217] |
| $I_2$ | $10^4$ | 3.0157 | $5.056 \times 10^{-2}$ | [2.9166, 3.1148] |

Table 1: Results of Importance sampling with standard normal as importance sampling distribution.

## b)

We will now discuss better choices for the importance sampling distributions.

To estimate $I_1$ we only take into account the samples which are greater than 2. Thus, by using a standard normal distribution as importance sampling distribution we loose a large proportion of the samples, which leads to an inefficient estimator. To address this issue we suggest a shifted exponential with support $x \in [2, \infty)$ as importance sampling distribution. With this importance sampling distribution we will only sample from $[2, \infty)$ and hence make use of every data point.

In order to estimate $I_2$ we require an importance sampling distribution with support including $(0, \infty)$. Furthermore, we want to consider the variance in the choice of the importance sampling distribution. We have that

$$Var\Big(\phi(Y)\frac{f(Y)}{g(Y)}\Big) = E\Big(\phi(X)^2\frac{f(X)}{g(X)}\Big) - E(\phi(X))^2.$$

Hence the ideal g(.) would be $g(x) = \frac{\phi(x)f(x)}{E(\phi(x))}$. However, as $E(\phi(x))$ is the unknown quantity we are trying to estimate, our best choice is to find g(.) such that $g(x) \propto \phi(x)f(x)$.
Substituting in what we know about $\phi(x)$ and f(x) we find that

$$\phi(x)f(x) = 3^2e^{-3x}$$

for $x \in (0, \infty)$. We note that this is proportional to the density function of an exponential random variable with parameter $\lambda = 3$. Since the exponential distribution has also support $[0, \infty)$ it also satisfies the requirement for the support of the importance sampling distribution. To conclude we suggest the use of an exponential distribution with parameter $\lambda = 3$ as importance sampling distribution to estimate $I_2$.

Finally, we note that in both cases $\frac{f(y)}{g(y)}$ is well behaved and does not get large for any y in the corresponding supports, hence we do not need to worry about tail behaviour.

In Figures 2 and 3 we plot $\phi(x)f(x)$ and the corresponding importance sampling distribution to check for proportionality. Note Figure 3 shows perfect proportionality.

In Table 2 we report the results obtained with the new choice of importance sampling distribution. We observe that the standard errors of our estimates for $I_1$ and $I_2$ have decreased by approximately 2 and 16 orders of magnitude respectively. This is a significant improvement, especially for the estimate of $I_2$.

| | n | Estimate | Standard Error | 95% Confidence Interval | Importance sampling distribution |
|---|---|---|---|---|---|
| $I_1$ | $10^4$ | 0.0174 | $2.462 \times 10^{-5}$ | [0.017, 0.0174] | Shifted exponential with $\lambda = 3$ |
| $I_2$ | $10^4$ | 3 | $5.66 \times 10^{-18}$ | [3, 3] | Exponential with $\lambda = 3$ |

Table 2: Results of Importance sampling with new choices for the importance sampling distributions.

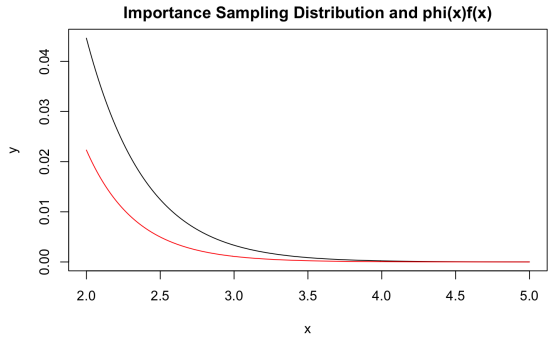


Figure 2: Plot of phi(x)f(x) and g(x) against x to check proportionality.

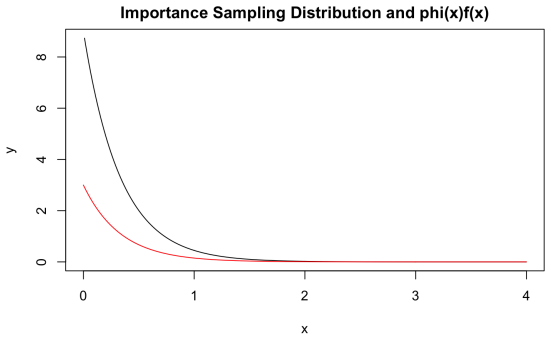


Figure 3: Plot of phi(x)f(x) and g(x) against x to check proportionality.

# Question 3

In this question we will consider the two-dimensional density

$$f(x,y) = \Big(\frac{2}{\pi}\Big)^{3/2} \frac{1}{\sigma\sqrt{3}} \frac{1}{\big(1+\frac{(x-y)^2}{3}\big)^2} exp\big(-\frac{1}{2\sigma^2}(x+y)^2\big),\ with \sigma = 5.$$

We will implement an importance sampler with importance sampling distribution X,Y$\sim$ N(0,1) independently to estimate $E((X+Y)^2)$. Finally, we will discuss the performance of this sampler possible improvements.

The implementation of the importance sampler is straight forward. We only note that, since X and Y are independent, the joint density of X and Y will be $g(x,y) = \phi(x)\phi(y)$ where $\phi(.)$ is the density function of a standard normal. It follows that the weights are $w_i = \frac{f(x_i,y_i)}{g(x_i,y_i)}$.

On the initial run with a sample size of $10^4$ for x and y we obtain an estimate of 23.88 with a large standard error of 17.48. This will need further investigation. We iterate the importance sampling process 20 times for a sample size of $10^4$ and report the estimates along with the standard error and the maximal weight each iteration in Table 3. We observe that whenever we have a large maximal weight the standard error becomes large. This has a significant impact on the estimates as they vary strongly. The large maximal weights indicate tail behaviour, which leads to a large influence of individual observation, and thus makes this importance sampler very unstable. The effect of the tail behaviour on the estimates produced with the importance sampler can also be observed in Figure 4 where we plot the estimates for increasing sample size n.

| Iteration | estimate | standard error | max(weights) |
|---|---|---|---|
| 1 | 23.88 | 17.48 | 4015.11 |
| 2 | 7.63 | 1.45 | 303.95 |
| 3 | 4.36 | 0.46 | 120.55 |
| 4 | 5.76 | 0.91 | 254.60 |
| 5 | 5.87 | 0.88 | 220.56 |
| 6 | 19.26 | 7.95 | 1604.93 |
| 7 | 13.15 | 4.77 | 1192.76 |
| 8 | 10.57 | 3.24 | 711.32 |
| 9 | 5.94 | 0.72 | 172.67 |
| 10 | 6.78 | 1.05 | 287.80 |
| 11 | 7.22 | 1.45 | 334.54 |
| 12 | 10.62 | 3.65 | 847.61 |
| 13 | 16.71 | 7.78 | 1991.22 |
| 14 | 9.15 | 3.26 | 927.88 |
| 15 | 10.00 | 2.83 | 616.67 |
| 16 | 5.64 | 0.77 | 182.75 |
| 17 | 6.71 | 1.06 | 271.26 |
| 18 | 5.36 | 0.54 | 103.38 |
| 19 | 5.50 | 0.65 | 139.37 |
| 20 | 15.33 | 7.65 | 1977.22 |

Table 3: Results of the importance sampler for multiple iterations with sample size $n = 10^4$ each iteration.

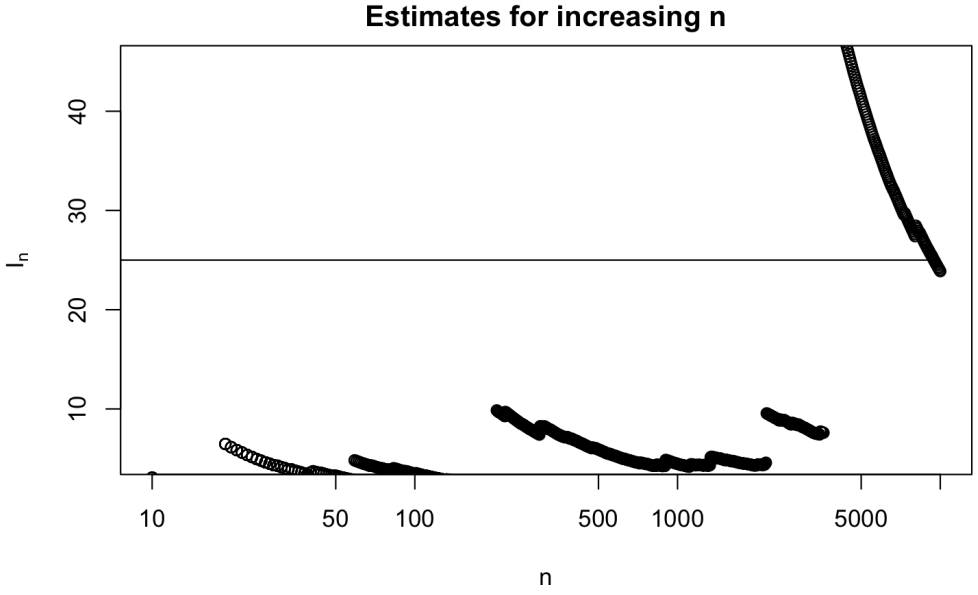


Figure 4: Estimates computed with the Importance sampler for increasing n. Note that the true value was computed to be approximately 25 using numerical integration.

To improve the performance of the importance sampler we need to control the tail behaviour. Since tail behaviour is caused by the importance sampling distribution having lighter tails than the target distribution, this can be achieved by choosing a distribution with heavier tails than the normal distribution. As the Cauchy distribution is known for its heavy tails, we suggest to use a Cauchy distribution as importance sampling distribution instead of the standard normal distribution. We report the corresponding results in Table 4. We find the maximal weight to be consistently around 50 and the standard error to be consistently around 2. The computed estimates are also much more stable. Overall this is a significant improvement to the algorithm using standard normal as importance sampling distribution.

| Iteration | estimate | std_err | max_weight |
|---|---|---|---|
| 1 | 24.17 | 2.01 | 50.80 |
| 2 | 24.11 | 1.99 | 49.79 |
| 3 | 26.41 | 2.26 | 49.92 |
| 4 | 23.83 | 1.91 | 47.73 |
| 5 | 21.08 | 1.71 | 51.74 |
| 6 | 24.58 | 2.07 | 49.90 |
| 7 | 26.25 | 2.28 | 52.21 |
| 8 | 23.38 | 2.02 | 52.44 |
| 9 | 20.70 | 1.81 | 51.91 |
| 10 | 27.26 | 2.31 | 52.98 |
| 11 | 26.87 | 2.20 | 52.98 |
| 12 | 22.11 | 1.93 | 53.00 |
| 13 | 24.41 | 2.10 | 52.79 |
| 14 | 27.17 | 2.33 | 52.15 |
| 15 | 27.95 | 2.22 | 51.08 |
| 16 | 26.23 | 2.19 | 50.69 |
| 17 | 22.29 | 1.93 | 52.13 |
| 18 | 22.18 | 1.93 | 51.96 |
| 19 | 23.06 | 1.87 | 48.11 |
| 20 | 21.71 | 1.76 | 49.32 |

Table 4: Results of the modified importance sampler with the Cauchy distribution as sampling distribution for a sample size of $n = 10^4$.

# Question 4

Suppose $X \sim Exp(2)$ and Y=X+10Z, where $Z \sim N(0,1)$. Furthermore, X and Z are independent.

## a)

In the following we derive the exact value of $E(X^2Y)$.

$$\begin{aligned} E(X^2Y) &= E(X^2(X+10Z)) && (1)\\ &= E(X^3+10ZX^2) && (2)\\ &= E(X^3)+E(10ZX^2) && (3)\\ &= E(X^3)+10E(Z)E(X^2) && (4)\\ &= E(X^3) && (5) \end{aligned}$$

Here we have used the linearity of expectation and the independence of X and Z in lines (3) and (4). Now integrating by parts once we obtain

$$E(X^3) = \int_0^\infty 3x^2 e^{-2x} dx = \frac{3}{2}E(X^2).$$

We now use the fact that Var(X)=$\frac{1}{4}$ and E(X)=$\frac{1}{2}$ when $X \sim Exp(2)$. Furthermore, using

$$Var(X) = E(X^2) - E(X)^2$$

we obtain E($X^2$)=$\frac{1}{2}$. Hence

$$E(X^2Y) = E(X^3) = \frac{3}{4}.$$

## b)

To estimate $E(XX^2Y)$ with Monte Carlo integration we first sample X and Z, then compute Y and finally compute the estimate $\hat{mc}_n = \sum_{i=1}^n X_i^2 Y_i$. With a sample of size $10^4$ we obtain an estimate of 1.015 which is significantly different from the true value 0.75.

## c)

We will now use Rao-Blackwellisation to obtain a better estimator. By conditional expectation we have $E(X^2Y) = E[E(X^2Y|X)]$. We use this to construct the rao-blackwellised estimator. In the following we compute $E(X^2Y|X)$:

$$\begin{aligned} E(X^2Y|X) &= X^2E(Y|X)\\ &= X^2E(X+10Z|X)\\ &= X^3+10X^2E(Z|X)\\ &= X^3. \end{aligned}$$

Thus, $E(X^2Y) = E(X^3)$. This leads us to the estimator $\hat{rb}_n = \sum_{i=1}^n X_i^3$ where $X_1, \ldots, X_n \sim Exp(2)$ independently. Computing an estimate on the same sample as in part b) (i.e. again n=$10^4$ and same seed) we obtain an estimate of 0.737, which is much closer to the true value.

## d)

Let us now compare the efficiency of the samplers constructed in b) and c). On a sample of size $10^5$ we obtain the following standard deviations of the estimates:

- Direct Monte Carlo Integration: 0.039
- After Rao-Blackwellisation: 0.01

We observe that we have managed to reduce the standard deviation by a factor of almost 4. Thus Rao-Blackwellisation yields a drastic improvement. To obtain such a standard deviation with the direct Monte Carlo sampler we would require a sample 16 times larger than that required for the rao-blackwellised sampler in part c).

# Question 5

In this question we will estimate

$$I_k = E[max(X_1^2, \ldots, X_k^2) - min(X_1^2, \ldots, X_k^2)]$$

where $X_1, \ldots, X_k \sim_{iid} N(0,1)$ and compare a numerical integration approach with a Monte-Carlo integration approach.

## a)

To compute the Monte-Carlo estimate, we sample k observations $10^4$ times from N(0,1) to obtain

$$\hat{I}_{i,k} = max(X_{i,1}^2, \ldots, X_{i,k}^2) - min(X_{i,1}^2, \ldots, X_{i,k}^2)$$

and then compute

$$\hat{I}_k = \frac{1}{10^4} \sum_{i=1}^{10^4} \hat{I}_{i,k}.$$

In Table  we report the estimate and standard error for each k.

| k | 1 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| Estimate | 0.0000 | 1.2728 | 2.3344 | 3.4255 | 4.5479 | 5.7359 |
| Standard Error | 0.0000 | 0.0151 | 0.0185 | 0.0207 | 0.0215 | 0.0226 |

Table 5: Estimates and Standard Errors with a sample size of $10^4$ for corresponding k.

## b)

We choose to use the midpoint formula as numerical integration method since it only requires one function evaluation for each point and has an error of $O(\frac{1}{n^2})$ as the trapezoidal formula. We first implement the function we want to integrate, namely

$$f(x_1, ..., x_k) = (max(x_1^2, ..., x_k^2) - min(x_1^2, ..., x_k^2)) \prod_{i=1}^{k} \phi(x_i)$$

where $\phi(.)$ is the density of the standard normal. We then use the expand.grid(.) function to obtain a grid of the equally spaced midpoints. Note that we restrict the range of integration to [-10,10] in each dimension and only allow $10^4$ function evaluations in total. We continue by evaluating f at each of the points of the grid. Finally we sum over all values and multiply by the corresponding weight. The weight for each point is $l^k$ where l corresponds to the distance between points in the same dimension and k to the dimension.

We obtain the results presented in Table 6. Note that the estimate for k=1 seems to be wrong, we should obtain 0 and hence bear in mind for the following discussion that the reported results might not be correct. For k=2 the numerical integration method seems to perform fine. However, for the higher dimensions it fails. This is consistent with the theory from lectures. The number of points on which we evaluate the function becomes too small (i.e. the grid distances become too large), which is why for higher dimensions Monte Carlo integration is really useful. From lectures we know that the error for the mid point rule applied to a d-dimensional integral is $O(\frac{1}{n^{2/d}}$, whereas the error of Monte Carlo integration is constant with $O(\frac{1}{\sqrt{n}})$. Finally, we note that for k=1 both methods should be exact and yield an estimate of 0.

| k | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Estimates | 0.1169757031 | 1.2689625492 | 0.5292057016 | 0.0003976486 |

Table 6: Results from the numerical integration with midpoint formula and using no more than $10^4$ function evaluations

# Appendix

Listing 1: Code for question 1

```r
# inversion for question 1
F <- function(x) (1-exp(-2*x))/(1-exp(-6))

rf <- function(n){
  # generate uniform on [0,1]
  U <- runif(n)
  # inverions and return X~f.
  return(-1/2*log(1-(1-exp(-6))*U))
}

set.seed(12345)
# compute empirical cdf
empirical <- ecdf(rf(1e3))
x <- seq(0,3,0.01)
plot(empirical, pch = 20, main='Empirical CDF and Thoretical CDF')
# line for true cdf
lines(x, F(x) , col='red')

# part b)
set.seed(123456)
# monte carlo integration
n <- 1e4
x <- rf(n)
mc <- mean(cos(x))
mc

#numerical integration
df <- function(x) 1/(1-exp(-6)) *2*exp(-2*x)
integrate(function(x) cos(x)*df(x), 0, 3)

# part c)

# 95% confidence interval
mc_sd <- sd(cos(x))/sqrt(n)
CI <- mc + mc_sd* qnorm(c(0.025, 0.975))
CI
# computations for length of the confidence interval
l <- 2*mc_sd*qnorm(0.975)
l
mc_sd
qnorm(0.975)
log10(10^(-6)/l)
```

Listing 2: Code for question 2

```r
set.seed(123456)
#sample y
n <- 1e4
y <- rnorm(n)
w <- dgamma(y, 2,3)/dnorm(y)
# compute estimate
I1 <- mean((y>2)*w)
I2 <- mean(1/y * w)
#standard error of is, NOTE ALREADY DIVIDED BY ROOT N.
alpha <-0.05

```

```
I1_sd <- sd((y>2)*w)/sqrt(n)
I1_CI <- I1 + I1_sd * qnorm(c(alpha/2, 1-alpha/2))

I2_sd <- sd(1/y*w)/sqrt(n)
I2_CI <- I2 + I2_sd * qnorm(c(alpha/2, 1-alpha/2))

cat('n', n ,':', 'est=', I1,' (', I1_sd,') ', 'CI=(', I1_CI,')','\n')
cat('n', n ,':', 'est=', I2,' (', I2_sd,') ', 'CI=(', I2_CI,')','\n')
df <- data.frame(n=rep(10^4,2), estimate = c(I1, I2), std_err = c(I1_sd,I2_sd), Conf_int=c(I1_CI,I2_CI))
xtable(df)

# part b)
# plot the targets to know what better importance sampling
x <- seq(0,4, 0.01)
fx <- 1/x * dgamma(x, 2,3)
plot(x,fx, col='white', main='Importance Sampling Distribution and phi(x)f(x)', ylab='y')
lines(x,fx)
lines(x,3*dexp(x, 3), col='red')

x <- seq(2,5, 0.01)
fx <- dgamma(x, 2,3)
plot(x,fx, col='white', main='Importance Sampling Distribution and phi(x)f(x)', ylab='y')
lines(x,fx)
lines(x,3*dexp(x, 3), col='red')

set.seed(123456)
#sample y
n <- 1e4
y1 <- rexp(n,3)+2
y2 <- y1-2
w1 <- dgamma(y1, 2,3)/dexp(y1-2,3)
w2 <- dgamma(y2, 2,3)/(dexp(y2,3))
# compute estimate
I1_e <- mean((y1>2)*w1)
I2_e <- mean(1/y2 * w2)
#standard error of is, NOTE ALREADY DIVIDED BY ROOT N.
alpha <-0.05

I1_sd_e <- sd((y1>2)*w1)/sqrt(n)
I1_CI_e <- I1_e + I1_sd_e * qnorm(c(alpha/2, 1-alpha/2))

I2_sd_e <- sd(1/y2*w2)/sqrt(n)
I2_CI_e <- I2_e + I2_sd_e * qnorm(c(alpha/2, 1-alpha/2))

cat('n', n ,':', 'est=', I1_e,' (', I1_sd_e,') ', 'CI=(', I1_CI_e,')','\n')
cat('n', n ,':', 'est=', I2_e,' (', I2_sd_e,') ', 'CI=(', I2_CI_e,')','\n')
```

Listing 3: Code for question 3

```
sigma <- 5
f <- function(x,y) (2/pi)^(3/2) * 1/(sigma*sqrt(3)) * 1/(1+(x-y)^2/3)^2 * exp(-1/(2*sigma^2)*(x+y)^2)

set.seed(123456)

n <- 10^4
alpha <- 0.05
```

```r
#sample y
x <- rnorm(n)
y <- rnorm(n)
w <- mapply(f, x, y)/(dnorm(x)*dnorm(y))
# compute estimate
is <- mean((x+y)^2*w)

#standard error of is, NOTE ALREADY DIVIDED BY ROOT N.
is_sd <- sd((x+y)^2*w)/sqrt(n)

#asmyptotic confidence interval
CI <- is + is_sd * qnorm(c(alpha/2, 1-alpha/2))

# check tail behaviour
set.seed(123456)
N <- 10^4
sample_size <- rep(N,20)
max_weights <- rep(0, 20)
estimates <- rep(0,20)
std_errors <- rep(0,20)
counter=1
for (n in rep(N,20)){
  x <- rnorm(n)
  y <- rnorm(n)
  w <- mapply(f, x, y)/(dnorm(x)*dnorm(y))
  cat('n=', n ,':', 'est=', mean((x+y)^2*w),' (', sd((x+y)^2*w)/sqrt(n),') ', 'max(↩
      w)=', max(w),'\n')

  max_weights[counter] <- max(w)
  estimates[counter] <- mean((x+y)^2*w)
  std_errors[counter] <- sd((x+y)^2*w)/sqrt(n)
  counter = counter +1

  # numerical integration to find exact value
  Int1 <- function(x) sapply(x, function(a1) integrate(function(y) (a1+y)^2*f(a1,y)↩
      ,-Inf,Inf)$value)

  integrate(Int1, -Inf, Inf)

  # plot for unstability
  set.seed(123456)

  n <- 10^4
  alpha <- 0.05
  #sample y
  x <- rnorm(n)
  y <- rnorm(n)
  w <- mapply(f, x, y)/(dnorm(x)*dnorm(y))

  In <- cumsum((x+y)^2*w)/1:n

  alln <- round(10^seq(1,4,length.out=1000)) # reduce number of data points
  # exact value for reference line
  exact <- 25

  plot(alln, In[alln], xlab='n', ylab=expression(I[n]), ylim=exact+c(-20,20), log='↩
      x',main='Estimates for increasing n')
  lines(c(1,n), c(exact,exact))


  # improvement using cauchy
```

```
  set.seed(123456)
  N <- 10^4
  sample_size <- rep(N,20)
  max_weights <- rep(0, 20)
  estimates <- rep(0,20)
  std_errors <- rep(0,20)
  counter=1
  for (n in rep(N,20)){
    x <- rcauchy(n)
    y <- rcauchy(n)
    w <- mapply(f, x, y)/(dcauchy(x)*dcauchy(y))
    cat('n=', n ,':', 'est=', mean((x+y)^2*w),' (', sd((x+y)^2*w)/sqrt(n),') ', '↩
        max(w)=', max(w),'\n')

    max_weights[counter] <- max(w)
    estimates[counter] <- mean((x+y)^2*w)
    std_errors[counter] <- sd((x+y)^2*w)/sqrt(n)
    counter = counter +1
  }
```

Listing 4: Code for question 4

```
# monte carlo integration
set.seed(123456)
n <- 10^4
X <- rexp(n,2)
Z <- rnorm(n)
Y <- X+10*Z

mc <- mean(X^2*Y)
mc

# monte carlo integration with rao-blackwellisation
set.seed(123456)
n <- 10^4
X <- rexp(n,2)

mc_rb <- mean(X^3)
mc_rb

# compare standard deviations
set.seed(123456)
n <- 10^5
X <- rexp(n,2)
Z <- rnorm(n)
Y <- X+10*Z

mc_sd <- sd(X^2*Y)/sqrt(n)
rb_sd <- sd(X^3)/sqrt(n)
```

Listing 5: Code for question 5

```
# part a)
phi <- function(x) max(x^2)-min(x^2)

genMCsamp <- function(k, nrep=1e4, r=function(n) rnorm(n)){
  I <- replicate(nrep, {
    x <- r(k)
    res=phi(x)
```

```
  })
  sterr <- sd(I)/sqrt(nrep);
  res <- c(That=mean(I), sterr=sterr)
  res
}

set.seed(123456)
k <- c(2^c(0:5))
Outputs <- sapply(k, genMCsamp)
Outputs

#part b)

#f evaluating the function to be integrated
f <- function(x)(max(x^2)-min(x^2))*prod(dnorm(x))

num_int <- function(k, nrep=10^4){

  n <- floor(nrep^(1/k))

  points <- seq(-10,10,length.out=n+1)
  midpoints <- points[1:n]+(points[2]-points[1])/2
  #list_mid <- replicate(k,midpoints)

  # expand.grid unfortunately does not work with list_mid hence need to adjust this manually
  if(k==1) grid <- midpoints
  if(k==2) grid <- expand.grid(midpoints,midpoints)
  if(k==4) grid <- expand.grid(midpoints,midpoints,midpoints, midpoints)
  if(k==8) grid <- expand.grid(midpoints,midpoints,midpoints, midpoints, midpoints,midpoints,midpoints, midpoints)

  if(k==1) sapply(midpoints, f)
  else y <- apply(grid, 1, f)

  w <- (points[2]-points[1])^k

  return(sum(y*w))
}

# computating the estimates
k_vec <- 2^(0:3)
sapply(k_vec, num_int)
```