Coursework - Computational Statistics

CID: 02091191

November 1, 2021

Abstract

In this document we are presenting the work conducted during the first Computational Statistics Coursework of the year.

1 First problem

(a) Estimation of constant k

Consider a probability density function f given by

$$f(x) = k\left(1+\cos(x)^3\right)\exp\left(-\frac{a_1+1}{10}x^2\right) \qquad \text{for } -1-a_2/2 \leq x \leq 1+a_2/2$$

As my CID is 02091191, we use as required $a_1 = 9$ and $a_2 = 1$, which allows to write f as

$$f(x) = k\left(1+\cos(x)^3\right)\exp\left(-x^2\right) \qquad \text{for } -3/2 \leq x \leq 3/2$$

and $f(x) = 0$ otherwise.
As f is a pdf, Kolmogorov's probability axioms require

$$\int_{\mathbb{R}} f(x)dx = 1 \quad \Leftrightarrow \quad k = \left[\int_{-\frac{3}{2}}^{\frac{3}{2}} \left(1+\cos(x)^3\right)\exp(-x^2)dx\right]^{-1}$$

In **R**, we use adaptive numerical integration implemented in `integrate()` function and get $k = 0.3576$. To check our value of k, we simply check that the integral of f equals to 1 as we now know k, which is the case here.

(b) Generate samples from f

As f is not a usual probability density function, we can't use pre-implemented functions in **R** to generate samples from f. However, we can do it by using acceptance-rejection algorithm whose full theory is explained in the lecture notes [1]. We want to sample a random variable with density f using another random variable that has density g such that there exists $C < \infty$:

$$\forall x,\ f(x) \leq Cg(x)$$

Assume g is the pdf of a normal distribution with mean 0 and variance $5(a_1+1) = 50$. The key in this algorithm is to find the appropriate constant C. We can calculate using simple upper bounds of cos and exp functions:

$$\frac{f(x)}{g(x)} = \sqrt{2\times 50\times\pi}k(1+\underbrace{\cos(x)^3)}_{\leqslant 1}\underbrace{\exp\left(-\frac{99}{100}x^2\right)}_{\leqslant 1}$$
$$\Rightarrow \frac{f(x)}{g(x)} \leqslant 2k\sqrt{100\pi} = C$$

In fact, we can even show that this upper bound C is the maximum of $\frac{f}{g}$ as the exponential factor maximises when $x = 0$ and equals to 1, and the cosinus factor maximises in every $x = 2p\pi, p \in \mathbb{N}$, especially in $x = 0$ and equals to 1.

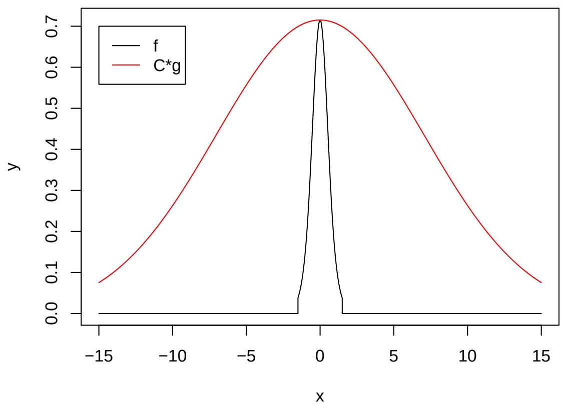


Figure 1.1: Probability density function f and function Cg

As shown on Figure 1.1, C is the optimal (the smallest) constant we could use as for the acceptance-rejection algorithm. We then implement the acceptance-rejection as described in the lecture notes [1] to generate 1000 samples from density function f.

In **R**, we compute the rate of acceptance of our acceptance-rejection algorithm: we get a rate of $\simeq 0.0785$. The theoretical rate of acceptance is $1/C \simeq 0.0789$: thus, our algorithm seems well implemented. To make sure the samples are following the distribution of f, we can plot the empirical cumulative distribution function compared to the theoretical one defined as: $F(x) = P(X < x) = \int_{-\infty}^{x} f(x)dx$. Computing F analytically may be long, so we approximate numerically this cumulative distribution function using the `integrate()` function in **R**.

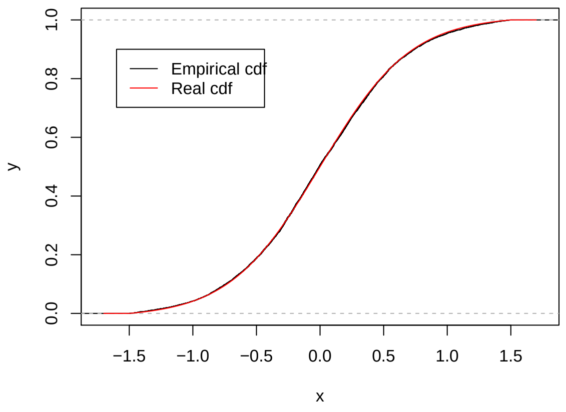


Figure 1.2: Comparison between empirical cdf (in black) and real cdf (in red)

Figure 1.2 shows that the 1000 samples from f generated via acceptance-rejection method have correctly been sampled as the empirical cdf in black is very close to the real cdf in red. Both of them are constant before -1.5 (equals to 0) and after 1.5 (equals to 1) as expected. We could do a Kolmogorov-Smirnov test in **R** to ensure the samples are correctly generated using both empirical and theoretical cdf, but our **R** code seems not able to do so.

(c) Monte-Carlo estimation of $\mathrm{E}[X^2]$

We now want to approximate $\mathbb{E}[X^2]$ using Monte-Carlo integration. We can write:

$$\mathbb{E}\left[X^2\right] = \int_{\mathbb{R}} x^2 f(x) dx = \int_{\mathbb{R}} h(x) f(x) dx \text{ where } h(x) = x^2$$

Then, Monte-Carlo approximation shows that:

$$\mathbb{E}\left[X^2\right] \simeq \frac{1}{n} \sum_{i=n}^{n} h\left(X_i\right) = \frac{1}{n} \sum_{i=0}^{n} X_i^2$$

where $X_1, \ldots, X_n$ are iid samples from density f. In **R**, we use our previous sample generator using acceptance-rejection algorithm to generate 10^4 iid samples from f, and compute the Monte-Carlo approximation. We obtain $\mathrm{E}[X^2] \simeq 0.3264$. Let $\hat{I}_n = \frac{1}{n} \sum_{i=n}^{n} h\left(X_i\right)$. Then by the central limit theorem:

$$\sqrt{n}\left(\hat{I}_n - \mathrm{E}(h(X))\right) \xrightarrow{d} N(0, \operatorname{Var}(h(X))) \quad (n \to \infty)$$

We can consistently approximate the unknown quantity $\operatorname{Var}(h(X))$ by the sample variance S^2. Hence, we can construct an asymptotic $1-\alpha$ confidence interval for $\mathrm{E}[X^2]$:

$$\left[\hat{I}_n + \frac{1}{\sqrt{n}} S z_{\alpha/2}, \hat{I}_n + \frac{1}{\sqrt{n}} S z_{1-\alpha/2}\right]$$

where z_β is such that $P(Z < z_\beta) = \beta$ for $Z \sim N(0,1)$. Thus, with $n = 10^4$, $\hat{I}_n = 0.3264$, $S = 0.4389$ and $z_{0.05} = -z_{0.95} = -1.6448$, we construct a 90% confidence interval for the Monte-Carlo estimate:

$$\hat{I}_n \in [0.3191,\ 0.3336] \text{ at a 90\% confidence level}$$

(d) Number of samples required for a given confidence interval length

If we want the length of the length interval at most equal 10^{-4}, we should have proceeded as follows. The length l of the confidence interval is proportional to $\sqrt{n}$. With the previous parameters, we obtained a length $l = 0.0144$. So if we want the length of the confidence interval to be 10^{-4}, we need a number of samples N which verifies:

$$l \frac{\sqrt{10^4}}{\sqrt{N}} = 10^{-4}$$

which gives

$$N = l^2 \times 10^{12} \simeq 14.4 \times 10^9$$

This N would require roughly 6 hours to compute the estimation.

2 Second problem

(a) Using numerical integration

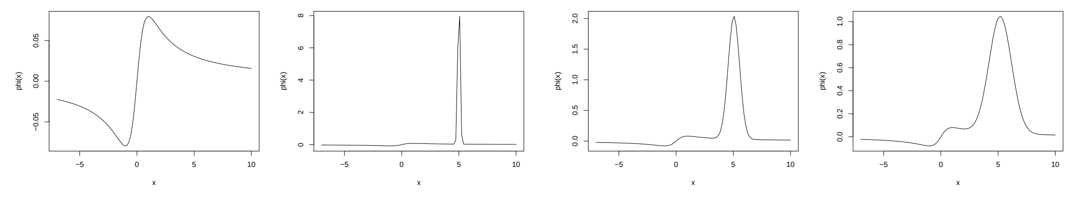
Suppose we are interested in estimating

$$I(\sigma) = \frac{1}{2} \int_{-\infty}^{\infty} x \left[\frac{1}{\pi\left(1+x^2\right)} + \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-5)^2}{2\sigma^2}\right)\right] dx \quad \text{for } \sigma \in \mathcal{S} := \{0.01, 0.1, 0.5, 1\}$$

In **R** we can use the `integrate()` function to compute $I(\sigma)$ for all values of σ. The `integrate()` function is adaptive, which means it will optimise the numerical grid according to the function's variations. We can re-write I as

$$I(\sigma) = \int_{-\infty}^{+\infty} \phi(x,\sigma) dx \text{ where } \phi(x,\sigma) = \frac{x}{2}\left[\frac{1}{\pi\left(1+x^2\right)} + \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-5)^2}{2\sigma^2}\right)\right]$$

We can hence plot the ϕ function for given values of σ as it is often useful before integrating.



(a) $\sigma = 0.01$ (b) $\sigma = 0.1$ (c) $\sigma = 0.5$ (d) $\sigma = 1$

Figure 2.1: ϕ functions plot for each value of σ

These plots on Figure 2.1 show how the value of σ change the shape of the ϕ function: this would have impacts on the integration using `integrate()` because if some variations are too small, the adaptive method may miss them and the estimation may be false.

	`integrate()`	mid-point
$\sigma = 0.01$	0	2.5
$\sigma = 0.1$	$4.1579e - 08$	2.5
$\sigma = 0.5$	2.5	2.5
$\sigma = 1$	2.5	2.5

Table 1: Comparison of I values for each value of σ obtained by `integrate()` function in **R** and by mid-point's method

Results in Table 1 show that the implemented `integrate()` function gives different results than a simple mid-point estimation. In fact, variations of ϕ are too small for $\sigma = 0.01, 0.1$ for the `integrate()` function to detect them. Hence, it considers the whole function flat and approximately equal to 0 and returns a value of 0. However, the results using the mid-point estimation seem more accurate. In fact, we could have noticed that:

$$I(\sigma) = \frac{1}{2}\left(\int_{-\infty}^{+\infty} x \frac{1}{\pi\left(1+x^2\right)} dx + \mathbb{E}[X]\right)$$

where X follows a normal distribution with mean 5 and variance σ^2, thus $\mathbb{E}[X] = 5$, and the left term is the "expectation" of a Cauchy distribution with location 0 and scale 1. However, a Cauchy distribution has no mean and variance, and thus we can't write the exact expectation. But a Cauchy distribution has a peak at its location value, which is 0 here. Hence, the integral I seems to be the mean between the peak values of a normal distribution with mean 5 and a Cauchy distribution with location 0. Then, this is why we obtain a value of I which seems not to depend on σ and is equal to $\frac{0+5}{2} = 2.5$.

(b) Using importance sampling

We can also use importance sampling to compute $I(\sigma)$. We sample from a normal distribution with mean 0 and variance 4. The importance sampling principle states that

$$\mathbb{E}[\psi(X)] = \mathbb{E}\left[\psi(Y)\frac{f(Y)}{g(Y)}\right]$$

where X has density $f = \phi$ here, $\psi(x) = \frac{x}{2}$, g is the density function of a normal distribution with mean 0 and variance 4 (the sampling distribution) and Y is sampled from g.

	Estimations
$\sigma = 0.01$	2.4094
$\sigma = 0.1$	2.3762
$\sigma = 0.5$	2.5250
$\sigma = 1$	2.5026

Table 2: $I(\sigma)$ values obtained using importance sampling

The estimations obtained with 10^4 samples are shown on Table 2. These estimations are close to the 2.5 value found using mid-point's method for numerical integration. This method seems more robust than the `integrate()` function for small values of σ.

(c) Using Monte-Carlo integration

Classic Monte-Carlo integration can also be used to estimate $I(\sigma)$. We have to notice that we are focusing on the expectation of a random variable $T = X + Y$ with X following a Cauchy distribution (location 0, scale 1) and Y following a Normal distribution (mean 5, variance σ^2). Hence, Monte-Carlo integration consists in generating $T_1, \dots, T_n$ observations from T and approximate

$$I(\sigma) \simeq \frac{1}{n}\sum_{i=1}^{n} T_i = \frac{1}{n}\sum_{i=1}^{n} X_i + Y_i \text{ where } X_1, \dots, X_n \overset{\text{iid}}{\sim} \text{Cauchy(0,1) and } Y_1, \dots, Y_n \overset{\text{iid}}{\sim} \text{Normal}(5,\sigma^2)$$

	Estimations
$\sigma = 0.01$	2.4082
$\sigma = 0.1$	2.2283
$\sigma = 0.5$	2.4203
$\sigma = 1$	2.7213

Table 3: $I(\sigma)$ values obtained using Monte-Carlo integration

Results shown in Table 3 show that the estimations are again close to the mid-point integration. However, we should expect a higher variance for these estimations than for the importance sampling estimations as the aim of importance sampling is to reduce variance. That is what we will work out next.

(d) Comparison of variance between importance sampling and Monte-Carlo integration

We want to compare the performance of both samplers studied in (b) and (c). We can compute in **R** the sampling errors of each estimator for both samplers, defined as $\frac{S}{\sqrt{n}}$ where S is the empirical standard deviation of samples and n the number of samples.

	Importance sampling	**Monte-Carlo**
$\sigma = 0.01$	0.49	14.63
$\sigma = 0.1$	0.24	18.87
$\sigma = 0.5$	0.14	2.92
$\sigma = 1$	0.13	11.09

Table 4: Comparison of sampling errors values for each value of σ obtained by importance sampling and Monte-Carlo integration

Results in Table 4 first show high sampling errors compared to the expected value (2.5), which means that both methods should be used with a "high" number of samples in order to get an accurate estimation of $I(\sigma)$. However, the importance sampling method has for all values of σ smaller sampling errors than for the Monte-Carlo estimations. Hence, we would prefer using the Importance Sampling method to approximate $I(\sigma)$. This is consistent as the aim of importance sampling is to reduce variance in our estimations using an appropriate sampling distribution.

3 Problem 3

Consider the two-dimensional density

$$f(x,y) = k\left(1+\frac{(x-2y)^2}{4}\right)^{-5/2}\left(1+\frac{(x+2y)^2}{3}\right)^{-2}$$

Assume we want to compute $\mathbb{E}[(X-Y)^2]$ using an importance sampler with sampling distribution $X, Y \sim \mathcal{N}(0,1)$. Thus, we define

$$\phi(x,y) = (x-y)^2 \text{ and } g(x,y) = g_X(x)g_Y(y) \quad \forall (x,y) \in \mathbb{R}^2$$

where $g_X = g_Y$ are defined as the pdf of a standard normal distribution (X and Y are in dependant thus the joint law is the product of both laws). Hence, the importance sampling principle method show that

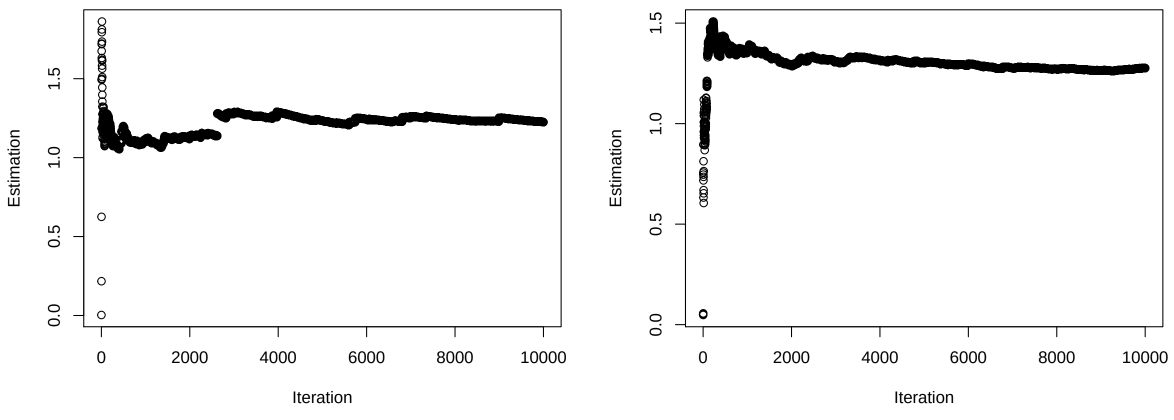
$$\mathbb{E}[\phi(X,Y)] = \mathbb{E}\left[\phi(U,V)\frac{f(U,V)}{g(U,V)}\right] \text{ where } U, V \sim \mathcal{N}(0,1) \text{ independently}$$

By implementing this algorithm in **R**, we obtain $\mathbb{E}[(X-Y)^2] \simeq 1.1848$ and a sampling error on this estimate of 0.0125. This seems to be a good estimation as the sampling error seems correct, but it may be improved. Indeed, importance sampling method depends on the choice of the sampling random variable and its pdf. Two conditions must be met when choosing this sampling probability density function g. First, we must have $\phi(x,y)f(x,y) > 0 \Rightarrow g(x,y) > 0$, which can be interpreted as the g function must have a domain containing the domain of f. Here, the domain is $\mathbb{R}^2$ so we have no choice and must choose a function g whose domain in $\mathbb{R}^2$ too. Then, we want the sampling density to have at least as heavy tails as f. If it is not the case, few generated samples may have large influence in the estimator and make it unstable (Lecture Notes [1]): this is known as tail behaviour. Previously, we sampled from a two-dimensional normal distribution. Thus, we could improve the estimation using a pdf having heavier tails than a normal distribution: we may think of a Student t distribution or a Cauchy distribution. Hence, we implement in **R** a new importance sampling estimation using these two distributions.

	Normal	**Student t**	**Cauchy**
Estimation	1.1848	1.3110	1.3109
Sampling error	0.0125	0.0093	0.0048

Table 5: Estimations and sampling errors of $\mathbb{E}[(X-Y)^2]$ via importance sampling with different sampling distributions | Normal: $\mathcal{N}(0,1)$, Student t: 2 degrees of freedom, Cauchy: location 0 and scale 1

The results in Table 5 show how the choice of the sampling distribution is crucial when using importance sampling. With a Cauchy distribution, we can obtain an estimation with a more than twice smaller sampling error than the previous normal distribution. This tail behaviour must me taken into account when choosing the sample distribution function. Here, we would prefer using a Cauchy distribution as the sample generating function as it is the one with the smallest sampling error. We chose Student t and Cauchy distribution because they are known to have heavier tails than a standard normal distribution.



(a) Sampling distribution: $\mathcal{N}(0,1)$ (b) Sampling distribution: t_2

Figure 3.1: Comparison of the estimation after each iteration between the previous normal sampling distribution and a t distribution

As shown on Figure 3.1, the normal sampling distribution may present jumps in the estimation (around 3000 iterations). This is because some few random samples may have a large weights in the estimation because the sampling distribution has thin tails. On the contrary, the t distribution used as sampling distribution seems more stable as there are not any jump in the estimation evolution. This is the result of choosing a sampling function with thicker tails.

4 Fourth problem

(a) Monte-Carlo integration

Suppose we are interested in estimating $I = \mathbb{E}[X\mathbb{1}\{X \geq 1\}]$ where $X \sim \mathcal{N}(0,1)$. We may first estimate I using simple Monte-Carlo integration.

$$I = \mathbb{E}[\phi(X)] \text{ where } \forall x \in \mathbb{R},\ \phi(x) = x\mathbb{1}\{x \geq 1\}$$

Hence

$$I \simeq \frac{1}{n}\sum_{i=1}^{n}\phi(X_i) = \sum_{i=1}^{n} X_i\mathbb{1}\{X_i \geq 1\} \text{ where } X_1,\dots,X_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$$

In **R**, we obtain $I \simeq 0.2450$ and a sampling error of 0.0059.

(b) Importance sampling

As in previous problems, we implement in **R** the importance sampling method in order to reduce the variance on the estimation. Here, a suitable sampling function would be a shifted exponential distribution with density $g(y) = \exp(-(y-1))$. This density is a good one for two reasons: first, it has the same support as f and then, an exponential distribution has heavier tails than a normal distribution. Then,

$$I = \mathbb{E}[\phi(X)] = \mathbb{E}\left[\phi(Y)\frac{f(Y)}{g(Y)}\right]$$

where Y is a random variable generated from density g (shifted exponential) defined above and f is the density function of X which is a standard normal density function. Hence

$$I \simeq \frac{1}{n}\sum_{i=1}^{n} \phi(Y_i)\frac{f(Y_i)}{g(Y_i)} \text{ where } Y_1, \dots, Y_n \overset{\text{iid}}{\sim} 1 + \text{Exp}(1)$$

In **R**, we obtain $I \simeq 0.2429$ and a sampling error of 0.001.

(c) Control variates

Let

$$U = X\mathbb{1}\{X \geq 1\}$$

We want to compute $\mathbb{E}[U]$. Then, let

$$V = \mathbb{1}\{X \geq 1\}$$

and assume that we know $\mathbb{E}[V] = \mathbb{E}[\mathbb{1}\{X \geq 1\}] = \mathbb{P}[X \geq 1]$. In fact, we can compute $\mathbb{P}[X \geq 1]$ in **R** using the `pnorm` function ($\mathbb{P}[X \geq 1] \simeq 0.1586$). Then, let

$$T = U + a(V - \mathbb{E}[V])$$

for some real constant a. We have $\mathbb{E}[T] = \mathbb{E}[U]$. Hence, we can compute $\mathbb{E}[T]$ in order to estimate $I = \mathbb{E}[U]$. We have

$$\mathbb{V}[T] = \mathbb{V}[U] + 2a\mathbb{C}\text{ov}[U, V] + a^2\mathbb{V}[V]$$

which is minimised for $a = -\frac{\mathbb{C}\text{ov}[U,V]}{\mathbb{V}[V]}$. In practice, a can't be computed directly so we use an estimation of covariance between U and V and an estimation of variance of V.

In **R**, we obtain $I \simeq 0.2408$ and a sampling error of 0.0018.

(d) Comparison between the three estimations

We can summarise the results obtained in all methods in the following table:

	Monte-Carlo	**Importance sampling**	**Control variates**
Estimation	0.2450	0.2429	0.2408
Sampling error	0.0059	0.0010	0.0018

Table 6: Estimations and sampling errors of I using Monte-Carlo integration, importance sampling and control variates methods

According to Table 6, we can first see that both reduction variance methods of importance sampling and control variates give smaller sampling errors than the classic Monte-Carlo integration. It seems that the importance sampling method is the most accurate one as it has the smallest sampling error: however, this is because we chose a wise sampling function g. If we had chosen another sampling function, we would have got different results. When a "good" sampling function is not easy to see, we could use the control variates technique in order to get a better estimation of I than a classic Monte-Carlo. Both importance sampling and control variates allow to reduce the sampling error compared to Monte-Carlo.

References

[1] Dr Sarah FILIPPI (Autumn 2021) *MATH70093 - Computational Statistics lecture notes*, Imperial College London MSc Statistics resources

[2] Kenneth LANGE (2010) *Numerical analysis for statisticians*, New York: Springer

[3] Marina EVANGELOU (Oct. 2021) Introduction to LaTeX

A R code

```
### Problem 1

set.seed(42)

a1 <- 9
a2 <- 1
g <- function(x) ifelse(x>=-1-a2/2 & x<=1+a2/2, (1+(cos(x))^3)*exp(-((a1+1)/10)*x^2), 0)
k <- 1/(integrate(g, -1.5, 1.5)$value)
f <- function(x) k*g(x)
x <- seq(-2, 2, by=0.001)
plot(x, f(x), type="l")

h <- function(x) dnorm(x, mean=0, sd=sqrt(50))
C <- sqrt(2*pi*50)*k*2
x <- seq(-15, 15, by=0.001)
plot(x, f(x), type="l", xlab="x", ylab="y")
lines(x, C*h(x), col="red")
legend(-15, 0.7, legend=c("f", "C*g"),
       col=c("black", "red"), lty=1:1)

acc.rej <- function(n, f, g, r){
  X <- c()
  cpt <- 0
  while(length(X) < n){
    cpt <- cpt + 1
    x <- r()
    u <- runif(1)
    if(u < f(x)/(C*g(x))){
      X <- c(X, x)
    }
  }
  return(list(data=X, rate=n/cpt))
}

result <- acc.rej(1000, f, h, r=function(x) rnorm(1, 0, sqrt(50)))
X <- result$data
rate <- result$rate

F <- function(t) integrate(f, lower=-1.7, upper=t)$value
x <- seq(-1.7, 1.7, by=0.001)
vec <- c()
for(elt in x){
  vec <- c(vec, F(elt))
}
plot(ecdf(X), main=NULL, ylab="y")
lines(x, vec, col="red")
legend(-1.6, 0.9, legend=c("Empirical cdf", "Real cdf"),
       col=c("black", "red"), lty=1:1)

set.seed(42)
```

```
time <- system.time(X <- acc.rej(1e4, f, h, r=function(x) rnorm(1, 0, sqrt(50)))$data)
MC <- mean(X^2)
S <- sd(X^2)

MC + S/sqrt(1e4) * qnorm(c(0.05, 0.95))
N/1e6*time[1]/3600


### Problem 2
sigma <- 0.5
phi <- function(x) 0.5*x*(dnorm(x,5,sigma)+dcauchy(x))
integrate(phi, -10, 10)

sigma <- 0.01
curve(phi, -7, 10, type="l")
sigma <- 0.1
curve(phi, -7, 10, type="l")
sigma <- 0.5
curve(phi, -7, 10, type="l")
sigma <- 1
curve(phi, -7, 10, type="l")

h <- 0.01
sigma <- 1
grid <- seq(-10, 10, h) + h/2
sum(phi(grid)) * h

psi <- function(x) x
f <- function(x) dcauchy(x)
g <- function(x) dnorm(x, 0, 4)
y <- rexp(10^4)
mean(phi(y)*f(y)/g(y))

set.seed(1320)
sigmas <- c(.01, .1, .5, 1)
x <- rnorm(1e4, 0, 4)
psi <- function(x) x
sapply(sigmas,
       function(s) mean(0.5*psi(x)*(dcauchy(x)+dnorm(x,5,s))/dnorm(x,0,4)))

seed(1331)
n <- 1e4
sapply(sigmas, function(s) mean(0.5 * (rcauchy(n) + rnorm(n, 5, sigma))))


set.seed(42)
x <- rnorm(1e4, 0, 4)
sapply(sigmas,
       function(s) sd(0.5*psi(x)*(dcauchy(x)+dnorm(x,5,s))/dnorm(x,0,4)))/sqrt(1e4)
sapply(sigmas,
       function(s) sd(0.5 * (rcauchy(n) + rnorm(n, 5, s))))/sqrt(1e4)
```

```
# Problem 3
set.seed(1521)
n <- 1e4
k <- 0.55135
f <- function(x,y) k * (1+((x-2*y)^2)/4)^(-5/2) * (1+((x+2*y)^2)/3)^(-2)
X <- rnorm(n)
Y <- rnorm(n)
mean(((X-Y)^2)*f(X,Y)/(dnorm(X)*dnorm(Y)))
sd(((X-Y)^2)*f(X,Y)/(dnorm(X)*dnorm(Y)))/sqrt(n)
plot(cumsum(((X-Y)^2)*f(X,Y)/(dnorm(X)*dnorm(Y)))/1:n, xlab="Iteration",
     ylab="Estimation")

U <- rt(n, 2)
V <- rt(n, 2)
mean(((U-V)^2)*f(U,V)/(dt(U, 2)*dt(V, 2)))
sd(((U-V)^2)*f(U,V)/(dt(U, 2)*dt(V, 2)))/sqrt(n)
plot(cumsum(((U-V)^2)*f(U,V)/(dt(U, 2)*dt(V, 2)))/1:n, xlab="Iteration",
     ylab="Estimation")

U <- rcauchy(n)
V <- rcauchy(n)
mean(((U-V)^2)*f(U,V)/(dcauchy(U)*dcauchy(V)))
sd(((U-V)^2)*f(U,V)/(dcauchy(U)*dcauchy(V)))/sqrt(n)
plot(cumsum(((U-V)^2)*f(U,V)/(dt(U, 2)*dt(V, 2)))/1:n, xlab="Iteration",
     ylab="Estimation")


### Problem 4
set.seed(1602)
n <- 1e4
X <- rnorm(n)

mean(ifelse(X>=1, X, 0))
sd(ifelse(X>=1, X, 0))/sqrt(n)

Y <- 1 + rexp(n)
mean(Y*(Y>1)*dnorm(Y)/dexp(Y-1))
sd(Y*(Y>1)*dnorm(Y)/dexp(Y-1))/sqrt(n)

U <- ifelse(X>=1, X, 0)
V <- (X>=1)
a <- - cov(U, V)/var(V)
T <- U + a*(V-pnorm(-1))
mean(T)
sd(T)/sqrt(n)
```