Assignment Week 4

Name: Ioannis Philippides

ULCN: s1579495

This assignment consists of two exercises both directed at computing (or approximating) the value of the integral

$$I = \frac{1}{\sqrt{2\pi}} \int_1^\infty x^2 e^{-x^2}\, dx.$$

Exercise 1. Importance sampling

(The first part is exercise 5.14 from Rizzo's book)

- Obtain a Monte Carlo estimate of I using importance sampling. (There are multiple solutions; make sure that your method is much more efficient than straightforward simulation from a normal distribution with mean zero.)

```
f <- function(x) { (1/sqrt(2*pi))*(x^2*exp(-x^2))*(x>1)} # set the function
# create empty vector of length 4 for estimation and standard error
SE <- est <- numeric(4)
M <- 10000 # 10000 samples per distribution
set.seed(1579495)

# use of Normal(1,1) to estimate g(x)
x <- rnorm(M, mean=1, sd=1)
g <- f(x) / dnorm(x, mean=1, sd=1)
est[1] <- mean(g)
SE[1] <- sd(g)

# use of exponential to estimate g(x):
x <- rexp(M) # default rate equals 1
g <- f(x) / dexp(x)
est[2] <- mean(g)
SE[2] <- sd(g)

#  use of gamma distribution:
x <- rgamma(M, shape=2, rate=1)
g <- f(x) / dgamma(x, shape=2, rate=0.9)
est[3] <- mean(g)
SE[3] <- sd(g)

# use of chisquare:
x <- rchisq(M, df=0.7) # set degrees of freedom equal 0.7
g <- f(x) / dchisq(x, df=0.7)
est[4] <- mean(g)
SE[4] <- sd(g)
```

distribution	Normal	exponential	gamma	chisquare
se	0.1340037	0.1688455	0.1512996	0.2581108
estimate	0.1029546	0.09873	0.1074385	0.0990978

We observe that chisquare distribution doesn't perform so well, comparing to the rest distributions. Normal distribution with mean and variance equal 1, has the smallest standard error.

Exercise 2. Numerical integration

- Write a function `trapeziumrule` that takes arguments `f, lower, upper, h`, equal to a function, lower and upper limits of the integral and a step size, that returns the integral of the function over the interval [`lower, upper`].

```
# function from integration.R file
trapeziumrule <- function(f, lower, upper, h) {
  x <- seq(lower, upper, h)
  y <- f(x)

  # the first and last one count only half
  total <- sum(y) - 0.5*y[1] - 0.5*y[length(y)]

  return(h*total)
}
```

- Use this function to find an approximation to the given integral I. Try the 4 step sizes $h = 0.01, 0.001, 0.0001, 0.0001$ and use an appropriate value for `upper`.

```
f <- function(x) { (1/sqrt(2*pi))*(x^2*exp(-x^2)) } # set the function
h <- c(0.01, 0.001, 0.0001, 0.00001) # create 4 steps for given h values
I <- numeric(4)
for (i in 1:4){
# As value for upper, 10 would be enough, because the (decreasing)
# function will be approximately zero
  I[i] <- trapeziumrule(f, lower = 1, upper = 10, h[i])
}
```

""	0.01	0.001	0.0001	0.00001
I	0.1011882	0.1011882	0.1011882	0.1011882

- Compute an approximation of I by using the `R` function `integrate`. Read the help page of the function to see how to handle the upper, infinite limit!

```
integrate(f, lower=1, upper=Inf)
```

```
## 0.1011882 with absolute error < 2.2e-08
```

```
# so we see that numerical integration, applying trapezium rule,
# is a pretty accurate method
```