Mandatory Assigment 2

Jone Damås - 256140

2022-10-03

Exercise 1:

a.)

$$F(x) = \int_{-\infty}^{x} f(t)dt = \int_{-\infty}^{x} \frac{1}{2}\exp(-|t|)dt$$

For x values less than 0, the integral from 0 to infinity isn't included.

$$F(x) = \frac{1}{2}\int_{-\infty}^{x} e^t\, dt = \frac{1}{2}e^t|_{-\infty}^{x} = \frac{1}{2}(e^x - e^{-\infty}) = \frac{1}{2}e^x, for \quad x < 0$$

For x values greater than 0, the integral from minus infinity to 0 has to be included.

$$F(x) = \frac{1}{2}\int_{-\infty}^{0} e^t\, dt + \frac{1}{2}\int_{0}^{x} e^{-t}\, dt = \frac{1}{2}e^t|_{-\infty}^{0} - \frac{1}{2}e^{-t}|_0^x = \frac{1}{2} + \frac{1}{2}(-e^x + e^0)$$
$$= \frac{1}{2}(2 - e^{-x}), for \quad x \geq 0$$

$$F(x) = \begin{cases} \frac{1}{2}e^x, & \text{if} \quad x < 0 \\ \frac{1}{2}(2 - e^{-x}) & \text{if} \quad x \geq 0 \end{cases}$$

The inverse of the cumulative distribution function:

$$u = \frac{1}{2}e^x \Rightarrow x = \log(2u)$$

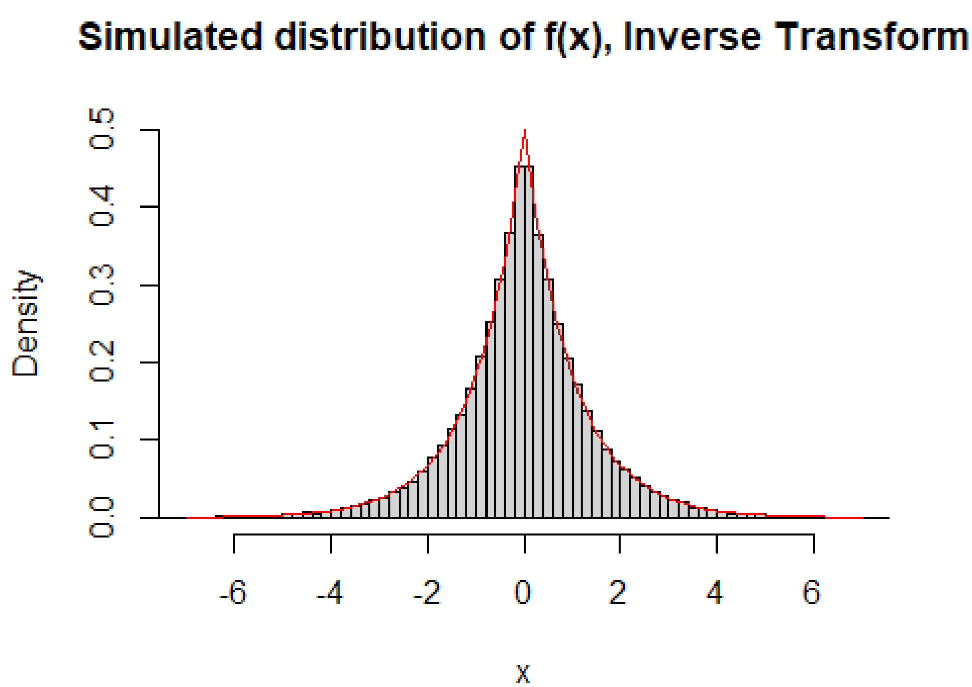
$$u = \frac{1}{2}(2 - e^{-x}) \Rightarrow e^{-x} = 2 - 2u \Rightarrow x = -\log(2 - 2u)$$

$$F^{-1}(u) = \begin{cases} \log(2u), & \text{if} \quad 0 < u \leq 1/2 \\ -\log(2 - 2u) & \text{if} \quad 1/2 < u < 1 \end{cases}$$

b.)

```
rm(list=ls())

# Function returning n random numbers with density f(x) by the inverse
transform method:
fx_pdf_IT <- function(n){
  # Generating n uniformly distrubuted random numbers between 0 and 1:
```

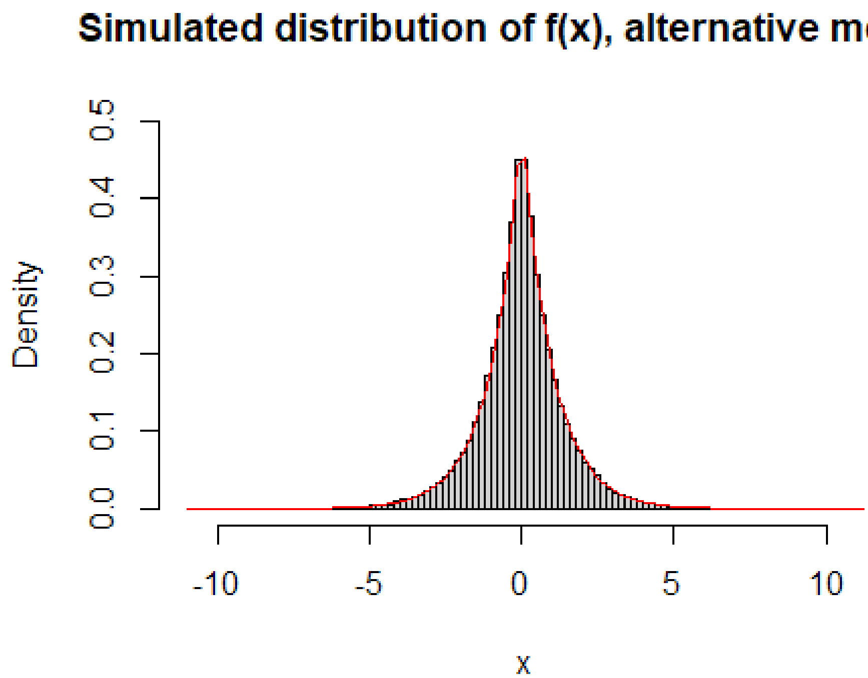
```
  u <- runif(n, 0, 1)
  # If/Else function returning correct transformed value considering if u is
less      than or greater than 1/2:
  return(ifelse(u<1/2, log(2*u) ,-log(2-2*u)))
}

# Histogram of the 100000 random numbers with density f(x):
hist(fx_pdf_IT(100000), probability=TRUE, ylim=c(0, 0.5), xlab='x',
main='Simulated distribution of f(x), Inverse Transform', breaks =100,
xlim=c(-7, 7))

# Add true density function:
x <- seq(-7, 7, 0.001)
curve(0.5*exp(-abs(x)), add=T, col='red')
```



c.)

```
fx_pdf_alt <- function(n){
  # Chose either -1 or 1 n times with equal probability (0.5):
  r <- sample(c(-1, 1), size=n, replace = TRUE)
  # Generate n random variables from an exponential distribution with
lambda=1:
  y <- rexp(n, 1)
  # Return r times y:
  return(r * y)
}
```

```
# Histogram plot of the results:
hist(fx_pdf_alt(100000), probability=TRUE, ylim=c(0, 0.5), xlab='x',
main='Simulated distribution of f(x), alternative method', breaks =100)

# Add true density function:
x <- seq(-7, 7, 0.001)
curve(0.5*exp(-abs(x)), add=T, col='red')
```



d.)

The proposal distribution with N(0, σ^2) is given by the pdf g(x):

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-0)^2}{2\sigma^2}\right)$$

We insert f(x) and g(x) into the accept-reject expression and simplify:

$$\frac{f(x)}{g(x)} \leq c \Rightarrow \frac{\sqrt{2\pi}\sigma \cdot \exp(-|x|)}{2 \cdot \exp(-x^2/2\sigma)} \leq c$$

$$\Rightarrow \frac{\sqrt{2\pi}\sigma}{2} \cdot \exp\left(-|x| + \frac{x^2}{2\sigma^2}\right) \leq c$$

We take the limit of the expression as c goes to infinity:

$$\lim_{x\to\infty} \frac{\sqrt{2\pi}\sigma}{2} \cdot \exp\left(-|x| + \frac{x^2}{2\sigma^2}\right) = \infty$$

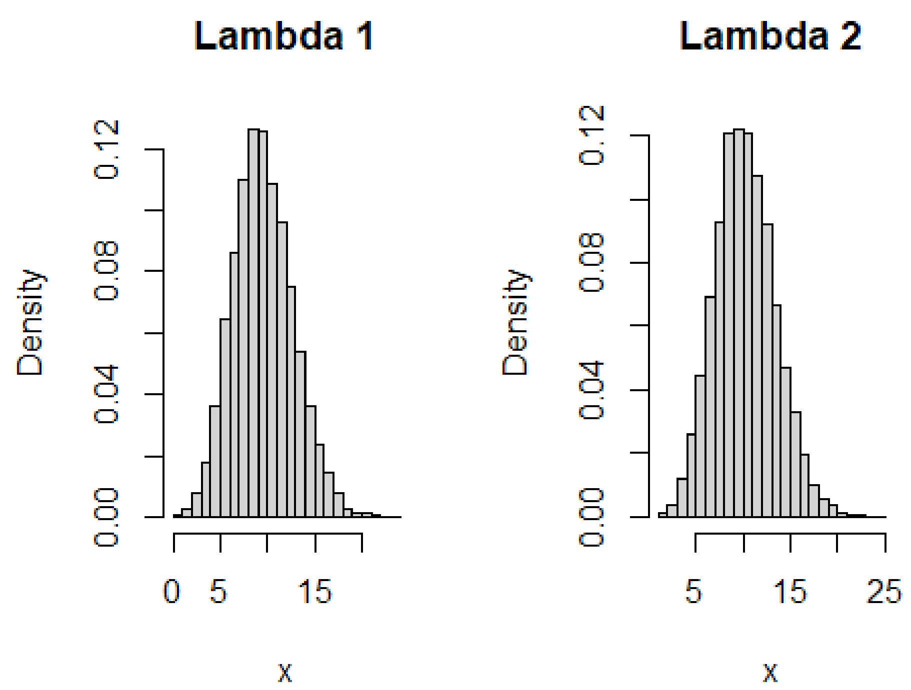
The expression diverges as x goes to infinity, and could therefore not be bound by a constant c. Therefore, a N(0, σ^2) distribution can not be used for an accept-reject method for f(x).

Exercise 2.)

a.)

```
# Intensity function 1:
lambda_1 <- function(t){
  return(1+0.5*cos(t * pi/5))
}

# Intensity function 2:
lambda_2 <- function(t){
  return(1+min(c(10, t))/6)
}

# non homogeneous poisson process simulation function:
NHPP <- function(a, b, lmax, lambda_f){
  if(max(lambda_f(seq(a,b,length.out = 100)))>lmax)
    stop("lmax < max(lambda_f)")
  else {
    # Number of simulations with:
    n_sim <- 3 * (b-a) * lmax

    # Times between events:
    t_between <- rexp(n_sim, lmax)
    t_cum <- a+cumsum(t_between)
    t_cum <- t_cum[t_cum<b]

    # Uniform number generation:
    u <- runif(length(t_cum))
    t_cum <- t_cum[u < lambda_f(t_cum)/lmax]
    return(t_cum)
  }
}

par(mfrow=c(1,2))
# Creating a poisson distribution of 10000 NHPP simulationss for lambda 1:
res_1 <- vector(length=10000)
for(i in 1:length(res_1)){
  res_1[i] <- length(NHPP(a=0,b=10,lmax=2,lambda_f=lambda_1))
}

hist(res_1, prob=T, breaks =20, main='Lambda 1', xlab='x')
cat('Expected value of N1(10):', mean(res_1), '\n')
```

```
## Expected value of N1(10): 10.0607

# Creating a poisson distribution of 10000 NHPP simulationss for lambda 2:
res_2 <- vector(length=10000)
for(i in 1:length(res_2)){
  res_2[i] <- length(NHPP(a=0,b=10,lmax=2,lambda_f=lambda_2))
}

hist(res_2, prob=T, breaks =20, main='Lambda 2', xlab='x')
```




```
cat('Expected value of N2(10):', mean(res_2), '\n')

## Expected value of N2(10): 10.7249
```

b.)

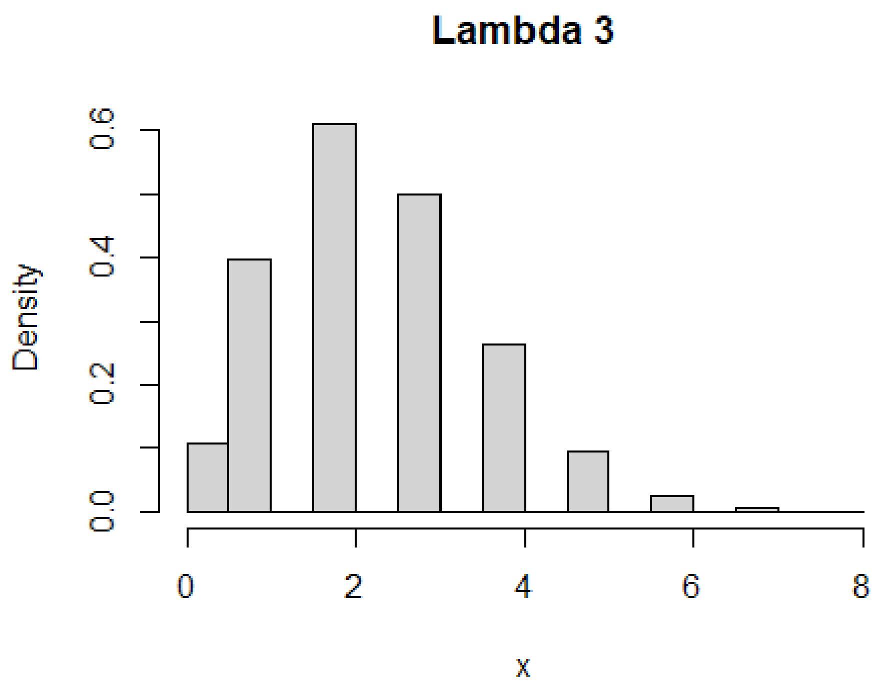
```
res_3 <- vector(length=10000)
for(i in 1:length(res_3)){
  res_3[i] <- length(NHPP(a=0,b=5,lmax=2,lambda_f=lambda_1))
}
cat('Expected value of N1(5):', mean(res_3), '\n')

## Expected value of N1(5): 5.0186

res_4 <- vector(length=10000)
for(i in 1:length(res_4)){
  res_4[i] <- length(NHPP(a=0,b=5,lmax=2,lambda_f=lambda_2))
}
```

```r
cat('Expected value of N2(5):', mean(res_4))

## Expected value of N2(5): 5.3932
```

c.)

d.)

```r
lambda_3 <- function(t){
  return(1+t/10)
}

NHPP_N3 <- function(b, lmax, lambda_f){
  if(max(lambda_f(seq(0, b, length.out = 100)))>lmax)
    stop("lmax < max(lambda_f)")
  else {
    # Number of simulations with:
    n_sim <- 3 * (b) * lmax

    # Generate event times from equation (2) with 0 as first entry:
    t_b <- vector(length = n_sim)
    t_b[1] <- 0.0
    for(i in 2:n_sim){
      t_b[i] <- sqrt(t_b[i-1]^2+20*rexp(1)+20*t_b[i-1]+100) - 10
    }

    # Times between events:
    t_cum <- cumsum(t_b)
    t_cum <- t_cum[t_cum<b]

    # Uniform number generation:
    u <- runif(length(t_cum))
    t_cum <- t_cum[u < lambda_f(t_cum)/lmax]
    return(t_cum)
  }
}

res <- vector(length=10000)
for(i in 1:length(res)){
  res[i] <- length(NHPP_N3(b=20, lmax=5, lambda_f=lambda_3))
}

hist(res, prob=T, breaks =15, main='Lambda 3', xlab='x')
```




```
cat('Expected value of N3(20):', mean(res), '\n')

## Expected value of N3(20): 2.4089
```

Exercise 3

a.)

```
AR_sim <- function(mu, sigma, phi, t, N){
  res_mat <- matrix(0.0, nrow=N, ncol=t)
  res_mat[,1] <- rnorm(N, mean=mu, sd=sigma)
  SD <- sqrt((1-phi^2)*sigma^2)

  for(n in 2:t){
    res_mat[,n] <- mu + phi*(res_mat[,n-1] - mu) + rnorm(N, mean=0.0, sd=SD)
  }
  return(res_mat)
}
```

b.)

```
T <- 1000
mu <- 0.0
sigma <- 1

phi_vals <- c(-0.8, 0.0, 0.8)

table_data <- matrix(nrow=3, ncol=3)
```

```r
for(i in 1:length(phi_vals)){
  XT <- AR_sim(mu, sigma, phi_vals[i], T, 10000)
  M1 <- 1/T*rowSums(XT)
  M2 <- 1/T*rowSums(XT)^2

  # Inserting phi values, mean of M1 and mean of M2 in a table:
  table_data[i ,1] <- format(phi_vals[i], digits=2)
  table_data[i ,2] <- format(mean(M1), digits=6)
  table_data[i ,3] <- format(mean(M2), digits=6)
}

colnames(table_data) = c('phi', 'mean', 'variance')

# Creating result table:
tbl <- as.table(table_data)
print(tbl)
```

```
##   phi  mean         variance
## A -0.8 -0.000165051 0.112354
## B 0    0.000416719  0.994959
## C 0.8  -0.00174727  9.23845
```

It does not seem like the differences in phi does a noteworthy effect on the mean of the realizations of the process, however, the variance increase as phi increases.

c.)

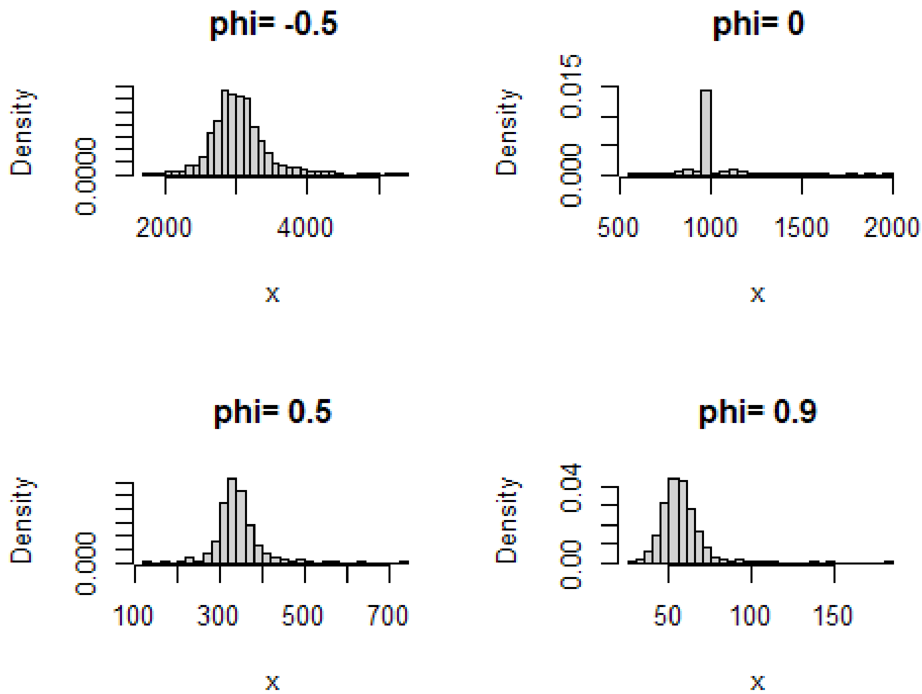
```r
# Excplicit ESS function:
explicit_ESS <- function(T, phi){
  return(T*(1-phi)/(phi+1))
}

# ESS function from the Mandatory assigment 2 pdf:
ESS <- function(x){
  return(as.numeric(coda::effectiveSize(x)))
}

t <- 1000
sims <- 1000

for(phi in c(0.98, -0.2)){
  # Vector for storing number of samples needed:
  n_vec <- vector(length = sims)
  for(i in 1:sims){
    var_value <- 10
    n <- 1
    # While loop expanding number of samples used for each iteration:
    while(var_value > 0.25^2){
      n = n + 1
      # AR simulation matrix:
      X <- AR_sim(0.0, 1, phi, t, n)
```

```
      # Calculate M1:
      M1 <- 1/T*rowSums(X)
      # Calculate variance of M1:
      var_value <- sd(M1)^2
    }
    # Storing number of samples needed:
    n_vec[i] <- n
  }
  print(paste('phi=', phi, 'Number of samples:', mean(n_vec)), quote = F)
}

## [1] phi= 0.98 Number of samples: 3.091
## [1] phi= -0.2 Number of samples: 2
```

It takes approximately 3 samples of M1 to get a Var(M1)<(0.25)^2 when phi = 0.98, and under 2 to get a Var(M1)<(0.25)^2 when phi = -0.2

d.)

```
phi_list <- c(-0.5, 0.0, 0.5, 0.9)
t <- 1000


t_data <- matrix(nrow=length(phi_list), ncol=4)

par(mfrow=c(2,2))
for(i in 1:length(phi_list)){

  # Calculate mean of all
  X <- AR_sim(0.0, 1, phi_list[i], t, 1000)

  # Applying ESS function to each row in the matrix, returning a vector with
the ESS of each row
  ESS_vec <- apply(X, 1, ESS)
  # Phi values:
  t_data[i, 1] <- format(phi_list[i], digits=2)
  # Explicit ESS:
  t_data[i, 2] <- format(explicit_ESS(t, phi_list[i]), digits=5)
  # ESS Estimator, taking the mean of the ESS vector:
  t_data[i, 3] <- format(mean(ESS_vec), digits=5)
  # Standard deviation:
  t_data[i, 4] <- format(sd(ESS_vec), digits=7)

  hist(ESS_vec, prob=T, xlab='x', main=paste('phi=', phi_list[i]), breaks =
30)
}
```




```
colnames(t_data) = c('phi', 'Explicit', 'ESS estimator mean', 'SD of
estimator')

# Creating result table:
tbl <- as.table(t_data)
print(tbl)

##   phi  Explicit ESS estimator mean SD of estimator
## A -0.5 3000     3038.8             412.194
## B 0    1000     1012.9             113.1773
## C 0.5  333.33   339.91             47.00623
## D 0.9  52.632   56.37              11.68121
```