Computational Statistics Lab 4

Hugo Morvan, Daniele Bozzoli

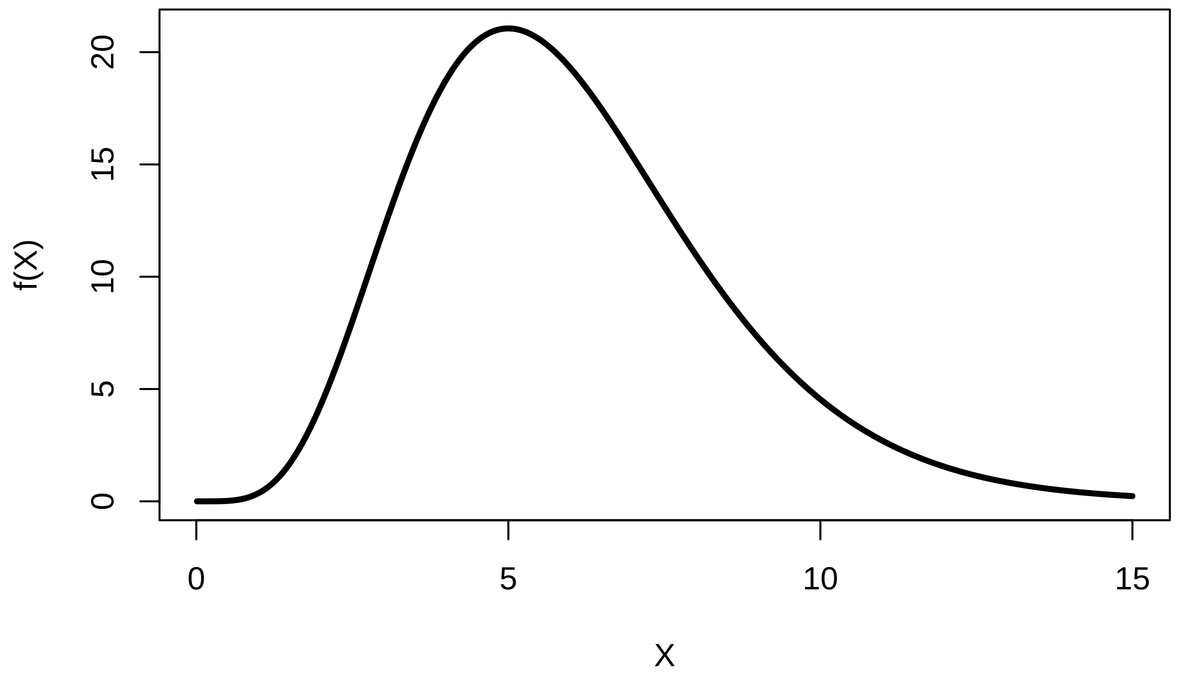
2023-11-28

Question 1: Computations with Metropolis-Hastings

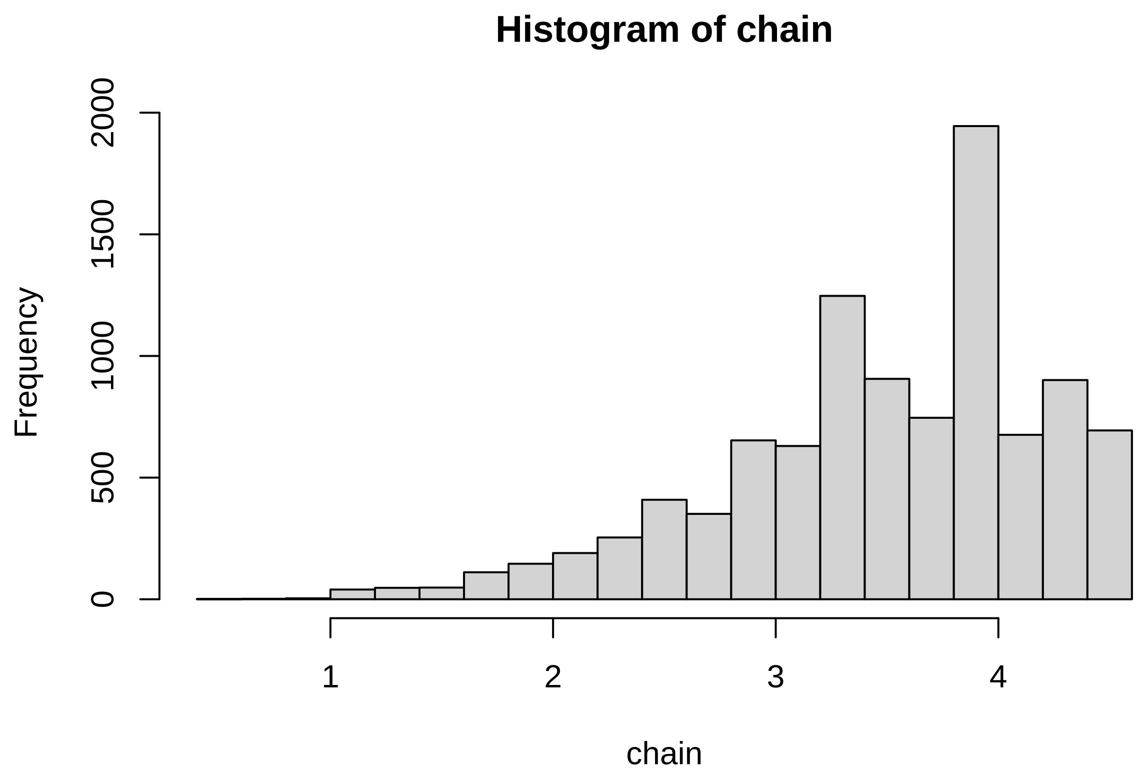
Consider a random variabel X with the following probability density function: $f(x) = x5e - x, x > 0$. The distribution is known up to some constant of proportionality. If you are interested (NOT part of the Lab) this constant can be found by applying integration by parts multiple times and equals 120.

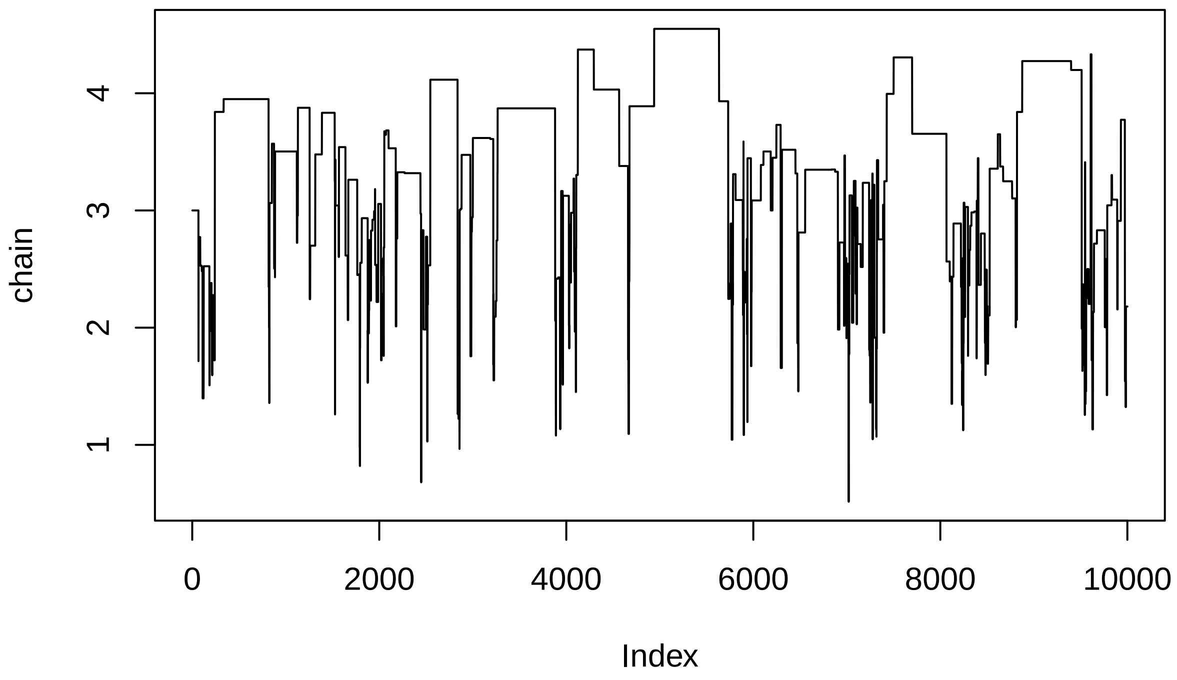
a.

Use the Metropolis–Hastings algorithm to generate 10000 samples from this distribution by using a log–normal LN (Xt , 1) proposal distribution; take some starting point. Plot the chain you obtained with iterations on the horizontal axis. What can you guess about the convergence of the chain? If there is a burn–in period, what can be the size of this period? What is the acceptance rate? Plot a histogram of the sample.



```
## acceptance rate = 0.0309
```



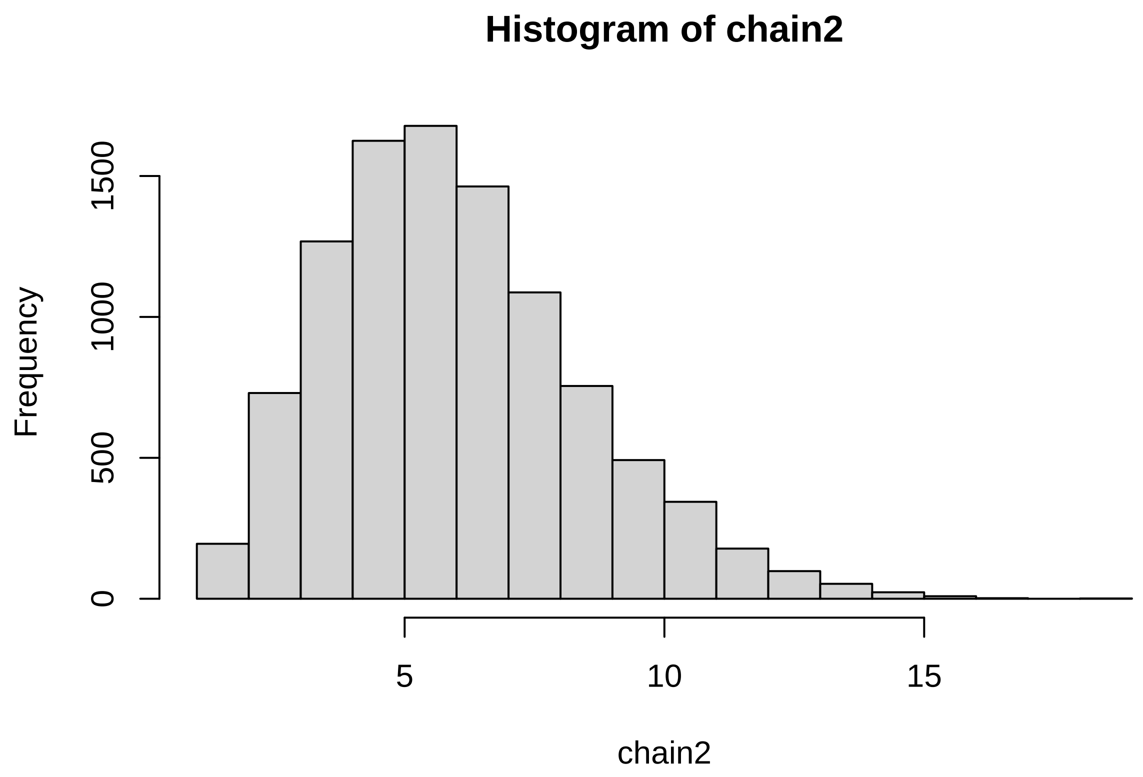


We observe from this plot that this method doesn't really converge and the majority of iterations are rejected (acceptance rate = 0.0309). Furthermore, a burn-in period is hard to detect / does not even look useful in this scenario, as this method is not really giving us good results overall.

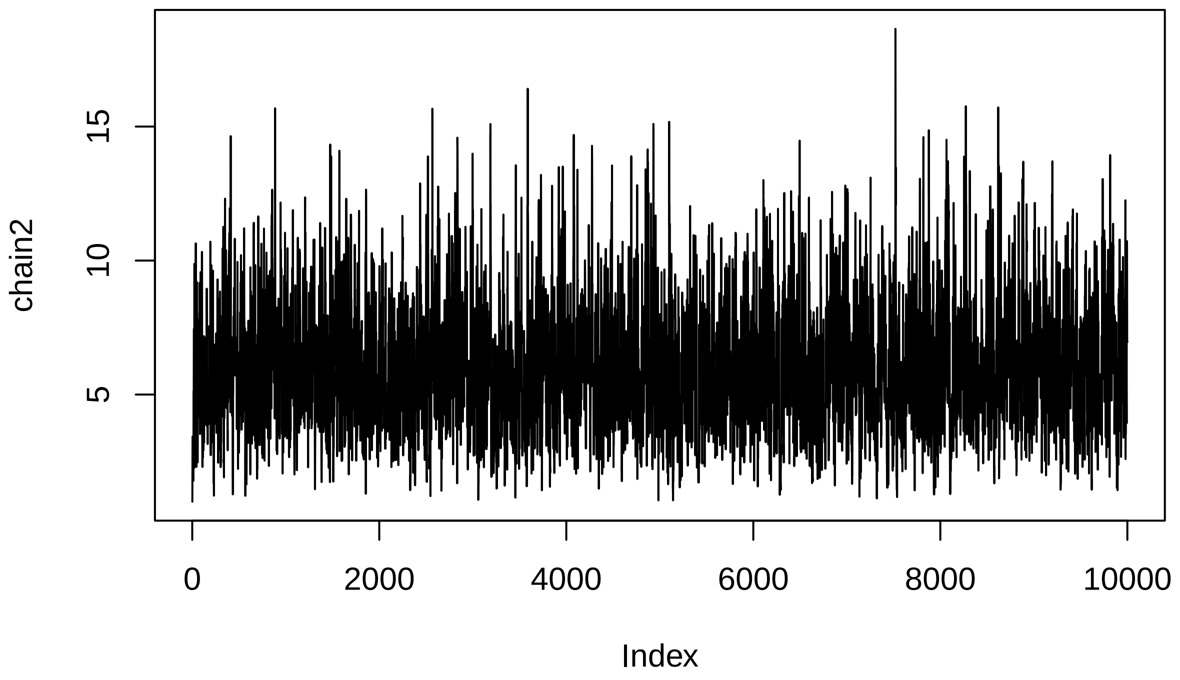
b.

Perform Part a by using the chi–square distribution as a proposal distribution, where is the floor function, meaning the integer part of x for positive x

```
## acceptance rate = 0.6048
```



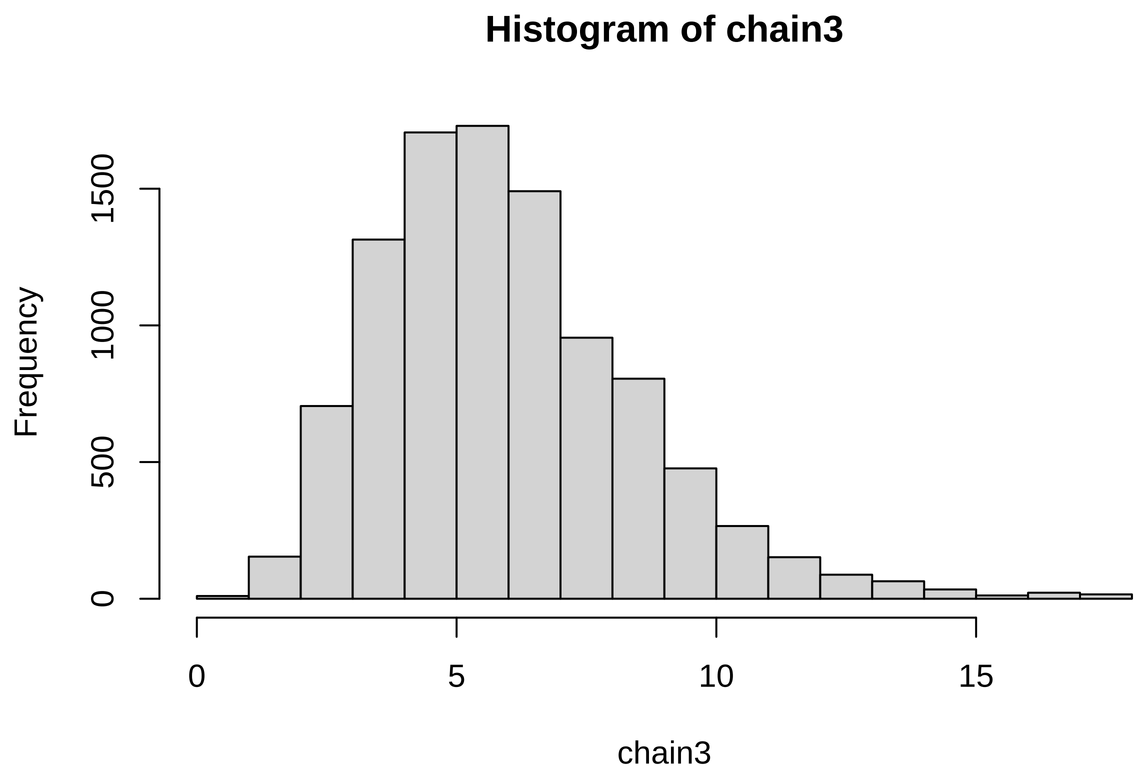
Histogram of chain2
Frequency
0
500
1000
1500
5
10
15
chain2

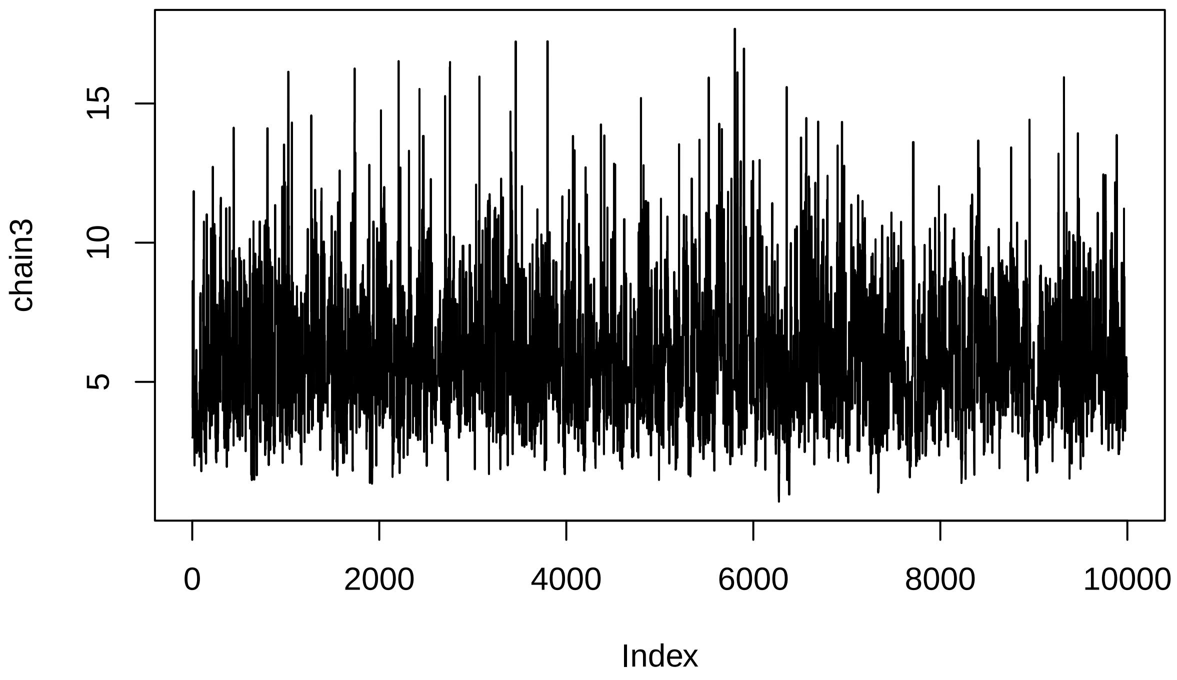


c.

Suggest another proposal distribution (can be a log normal or chi–square distribution with other parameters or another distribution) with the potential to generate a good sample. Perform part a with this distribution.

```
## acceptance rate = 0.3933
```

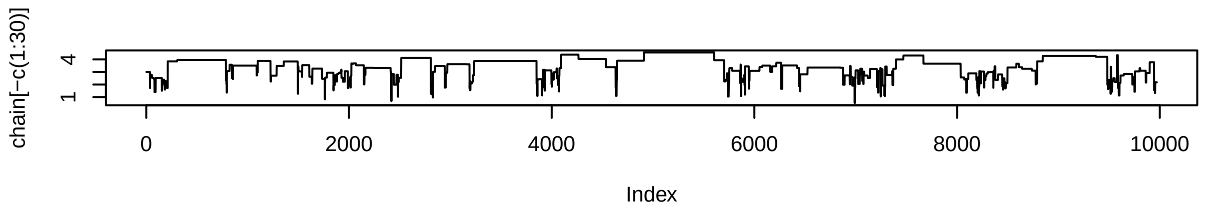

Histogram of chain3
Frequency
0
500
1000
1500
0
5
10
15
chain3

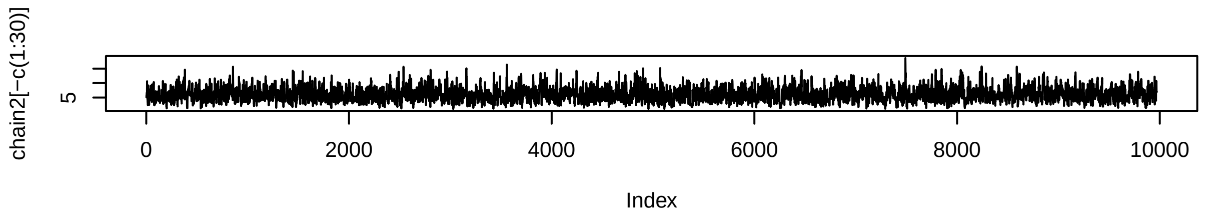


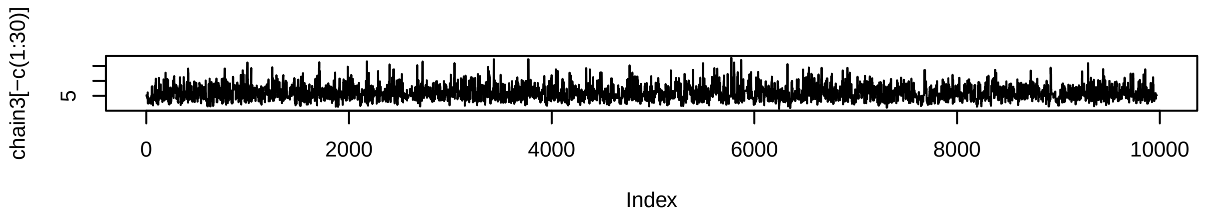
d.

Compare the results of Parts a, b, and c and make conclusions.

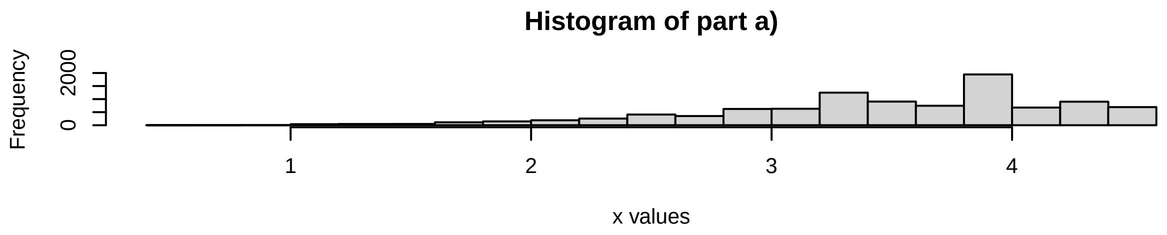
Looking at the histograms, it is clear how the second and third methods generate more reasonable results, both with better acceptance rates than the first method. The first method seems to generate too many samples with low values (< 4). The other two seem to generate fair results, compared to the distribution of f(x). They produce very similar results with slightly different acceptance rates, the Chi Squared distribution seems to be the best proposal distribution to choose for our case. A burn-in period of 30 was chosen (mainly for method b) and c) as method a) does not get any better even with this burn-in period).

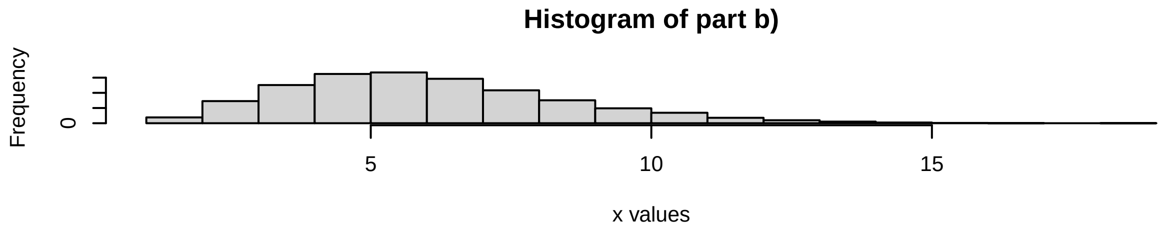


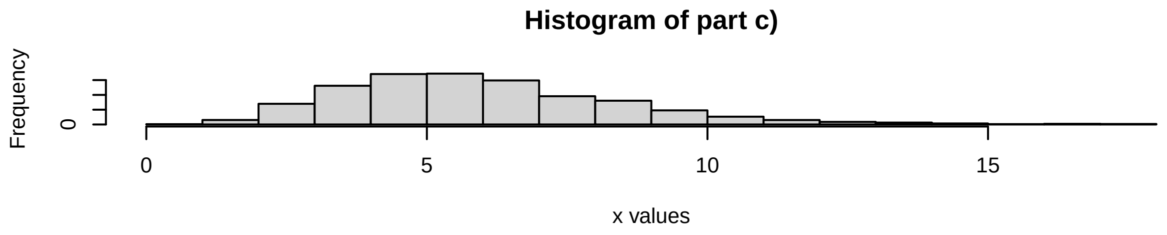


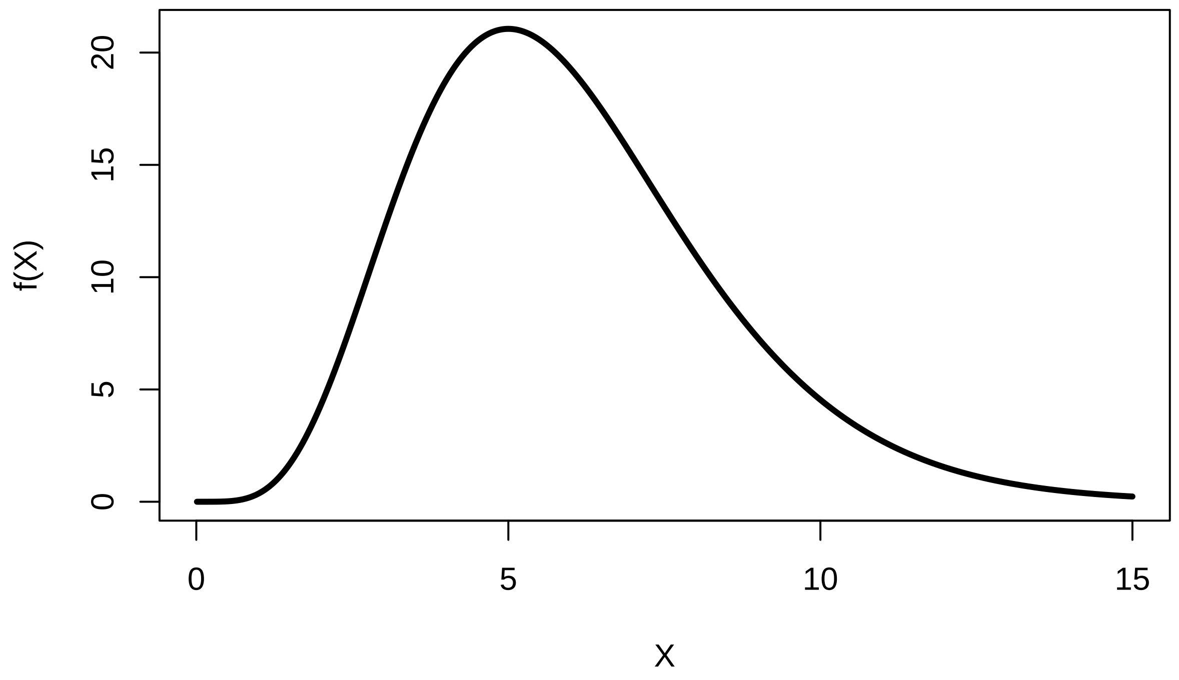


```
## Method a) acceptance rate = 0.0309

## Method b) acceptance rate = 0.6048

## Method c) acceptance rate = 0.3933
```


Histogram of part a)
Frequency
0
2000
1
2
3
4
x values


Histogram of part b)
Frequency
0
5
10
15
x values


Histogram of part c)
Frequency
0
0
5
10
15
x values



e.

Estimate

$$E(X) = \int_0^\infty x f(x) dx$$

using the samples from Parts a, b, and c.

```
## the expected value of the first method is: 3.48766
```

```
## the expected value of the second method is: 6.005467
```

```
## the expected value of the third method is: 5.975368
```

f.

The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

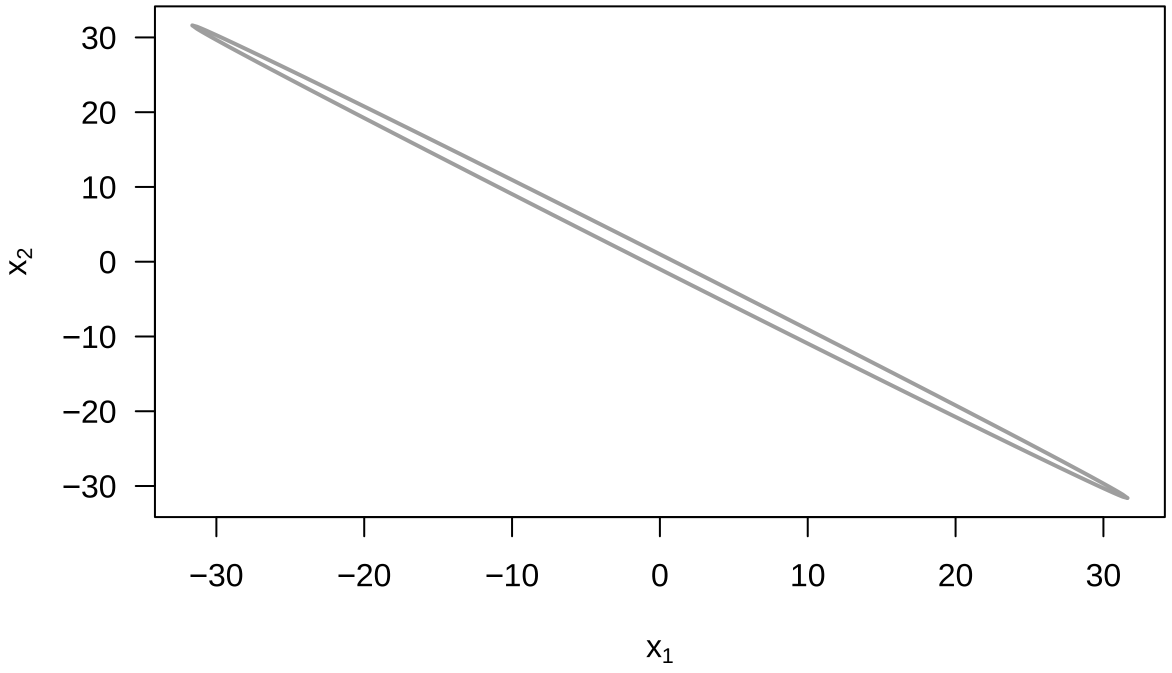
The distribution f(x) is a Gamma distribution with parameters alpha=6 and beta=1, hence E(x) = alpha/beta = 6. The results we obtained are pretty accurate with b) and c) methods, but not accurate at all using the first method a).

Question 2: Gibbs sampling

Let X = (X1 , X2) be a bivariate distribution with density $f(x_1, x_2) = 1\{x_1^2 + wx_1x_2 + x_2^2 < 1\}$ for some specific w with |w| < 2. X has a uniform distribution on some two-dimensional region. We consider here the case w = 1.999 (in Lecture 4, the case w = 1.8 was shown).

a.

Draw the boundaries of the region where X has a uniform distribution. You can use the code provided on the course homepage and adjust it.



b.

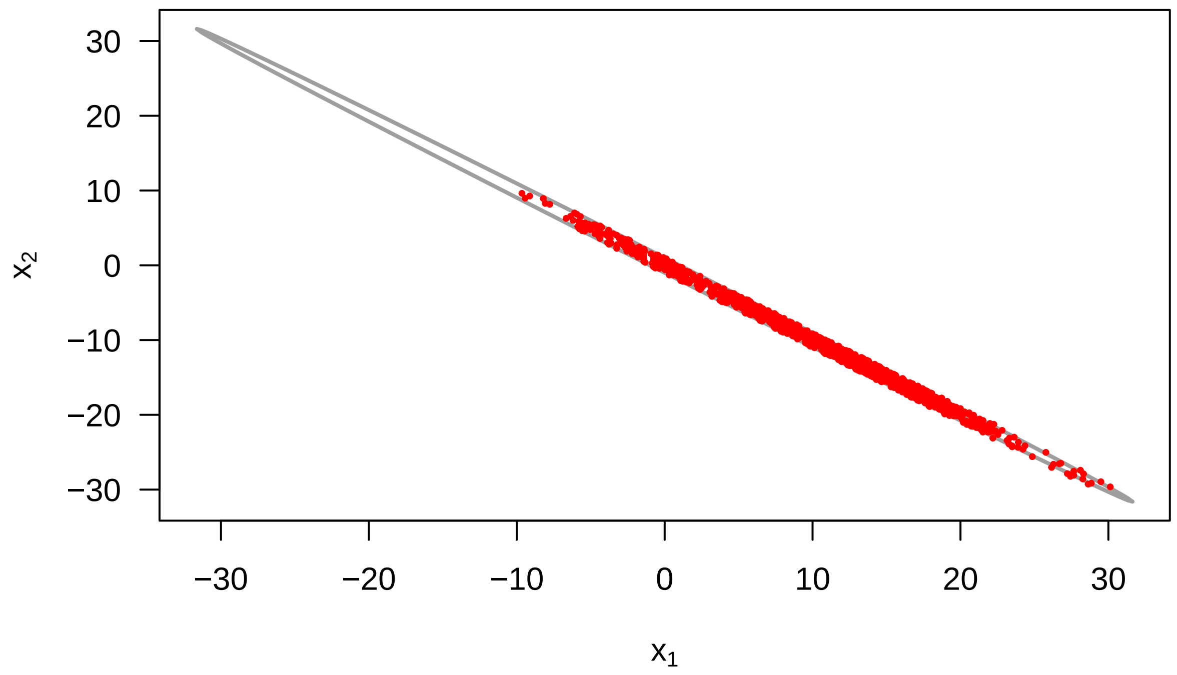
What is the conditional distribution of X1 given X2 and that of X2 given X1 ?

The conditional distribution $f(x_1|x_2)$ is a uniform distribution on the interval $(-0.9995x_1 - \sqrt{1 - 0.00099975x_1}, -0.9995x_1 + \sqrt{1 - 0.00099975x_1})$

The conditional distribution $f(x_2|x_1)$ is a uniform distribution on the interval $(-0.9995x_2 - \sqrt{1 - 0.00099975x_2}, -0.9995x_2 + \sqrt{1 - 0.00099975x_2})$

c.

Write your own code for Gibbs sampling the distribution. Run it to generate n = 1000 random vectors and plot them into the picture from Part a.



Determine P (X1 > 0) based on the sample and repeat this a few times (you need not to plot the repetitions). What should be the true result for this probability?

P(X1 > 0) = 0.87. The true result should be 0.5. Using the same code, we can repeat the experiment a few times and it seems that the result is always close to 0.5, but this is very dependent on the starting values of X1 and X2.

After running the code 100 times, we get a mean of 0.47483.

d.

Discuss, why the Gibbs sampling for this situation seems to be less successful for w = 1.999 compared to the case w = 1.8 from the lecture.

The Gibbs sampling for this situation seems to be less successful for w = 1.999 compared to the case w = 1.8 from the lecture because the boundaries of the region where X has a uniform distribution are very close to each other. This makes it hard for the simulation to cover the whole region because the probability of the simulation to jump from one side of the region to the other is very low. This is why the simulation is very dependent on the starting values of X1 and X2.

e.

We might transform the variable X and generate $U = (U_1, U_2) = (X_1 - X_2, X_1 + X_2)$ instead. In this case, the density of the transformed variable $U = (U_1, U_2)$ is again a uniform distribution on a transformed region (no proof necessary for this claim). Determine the boundaries of the transformed region where U has a uniform distribution on. You can use that the transformation corresponds to $X_1 = (U_2 + U_1)/2$ and $X_2 = (U_2 - U_1)/2$

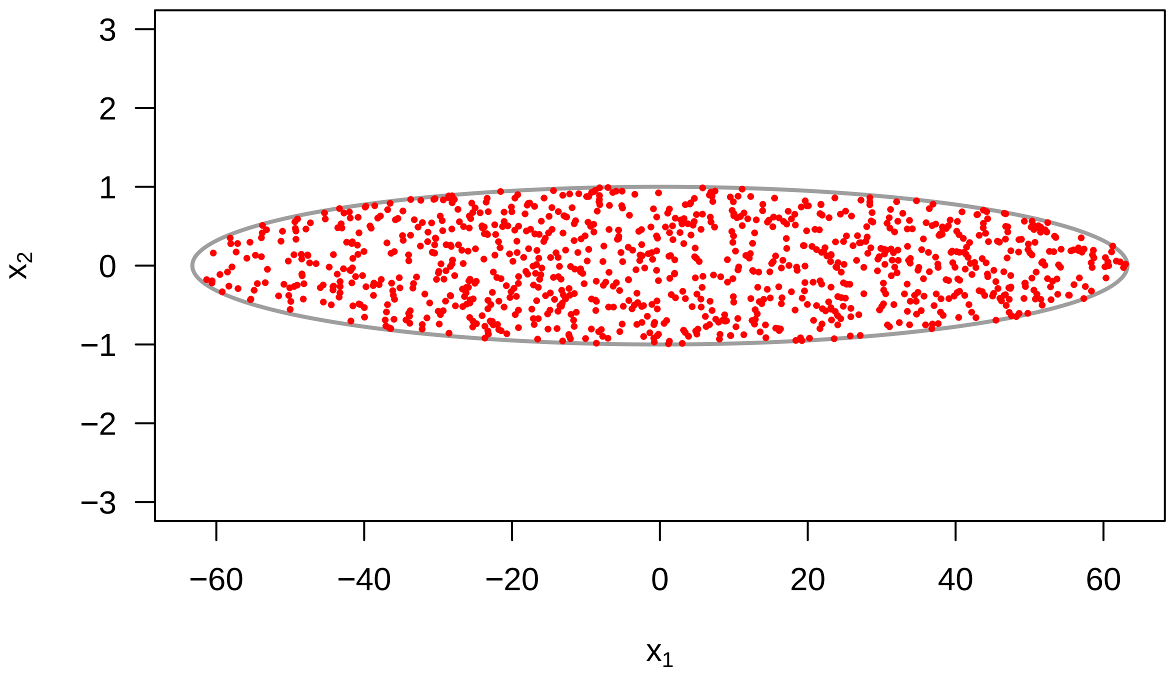
and set this into the boundaries in terms of X_i . Plot the boundaries for (U_1, U_2). Generate n = 1000 random vectors with Gibbs sampling for U and plot them. Determine $P(X_1 > 0) = P((U_2 + U_1)/2 > 0)$. Compare the results with Part c.

Using the transformation, we get the following expression for $f(u_1, u_2)$:

$f(u_1, u_2) = 1\{u_2^2(0.5 + 0.25 * w) + u_1^2(0.5 - 0.25 * w) < 1\}$

we can then get the conditional by solving the equation $u_2^2(0.5 + 0.25 * w) + u_1^2(0.5 - 0.25 * w) - 1 = 0$ for u_1 and u_2 respectively.

We then obtain that the conditional distribution for u_2 given u_1 is a uniform distribution on the interval $(-\sqrt{\frac{1-u_1^2(0.5-0.25*w)}{0.5+0.25*w}} , \sqrt{\frac{1-u_1^2(0.5-0.25*w)}{0.5+0.25*w}})$ and the conditional distribution for u_1 given u_2 is a uniform distribution on the interval $(-\sqrt{\frac{1-u_2^2(0.5+0.25*w)}{0.5-0.25*w}} , \sqrt{\frac{1-u_2^2(0.5+0.25*w)}{0.5-0.25*w}})$ and



The mean of the 100 simulations is 0.49937. From the plot, we can see that the simulation is more successful than the one in part c. because it is less dependent on the starting values of U1 and U2 and the vector of U1 and U2 covers the whole region.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
### a)

f <- function(x){
  stopifnot(x>0)
  return(x^5*exp(-x))
}

X <- seq(0.01,15,0.01)
plot(X, f(X), type="l", lwd=3)      # f(x) distribution

starting_point <- 3
N <- 1E4
set.seed(12345)

proposal_dist <- function(Xt){
  return(rlnorm(1, Xt, 1))
}

chain <- starting_point
accept <- 0

for (i in 1:N){

  x <- chain[length(chain)]
  # candidate point x_star from g
  x_star <- proposal_dist(x)

  # MH ratio
  R     <- (f(x_star) * dlnorm(x, x_star, 1)) /
            (f(x) * dlnorm(x_star, x, 1))
  ap    <- runif(1)
  if (ap < R){
    chain <- c(chain, x_star)
    accept <- accept + 1
  } else chain <- c(chain, x)
}

cat("acceptance rate =", accept / N)
hist(chain)
plot(chain, type="l")

#---------------------------------------------
### b)

starting_point2 <- 1
N <- 1E4
set.seed(12345)

proposal_dist2 <- function(Xt){
  return(rchisq(1, floor(Xt+1)))
}

chain2 <- starting_point2
accept2 <- 0

for (i in 1:N){
  x <- chain2[length(chain2)]
  # candidate point x_star from g
  x_star <- proposal_dist2(x)

  # MH ratio
  R     <- (f(x_star) * dchisq(x, floor(x_star + 1))) /
    (f(x) * dchisq(x_star, floor(x+1)))
  ap    <- runif(1)
  if (ap < R){
    chain2 <- c(chain2, x_star)
    accept2 <- accept2 + 1
  } else chain2 <- c(chain2, x)
}

cat("acceptance rate =", accept2 / N)
hist(chain2)
plot(chain2, type="l")
#---------------------------------------------
### c)

starting_point3 <- 3
N <- 1E4
set.seed(12345)

proposal_dist3 <- function(Xt){
  return(rexp(1, Xt^-1))
}

chain3 <- starting_point3
accept3 <- 0

for (i in 1:N){

  x <- chain3[length(chain3)]
  # candidate point x_star from g
  x_star <- proposal_dist3(x)

  # MH ratio
  R     <- (f(x_star) * dexp(x, x_star^-1)) /
    (f(x) * dexp(x_star, x^-1))
  ap    <- runif(1)
  if (ap < R){
    chain3 <- c(chain3, x_star)
    accept3 <- accept3 + 1
  } else chain3 <- c(chain3, x)
}

cat("acceptance rate =", accept3 / N)
hist(chain3)
plot(chain3, type="l")
#---------------------------------------------
### d)
```

```r
par(mfrow=c(3,1))
plot(chain[-c(1:30)], type="l")
plot(chain2[-c(1:30)], type="l")
plot(chain3[-c(1:30)], type="l")

cat("Method a) acceptance rate =", accept / N, "\n")
cat("Method b) acceptance rate =", accept2 / N, "\n")
cat("Method c) acceptance rate =", accept3 / N, "\n")

par(mfrow=c(3,1))
hist(chain[-c(1:30)], main = "Histogram of part a)", xlab = "x values")
hist(chain2[-c(1:30)], main = "Histogram of part b)", xlab = "x values")
hist(chain3[-c(1:30)], main = "Histogram of part c)", xlab = "x values")

par(mfrow=c(1,1))
plot(X, f(X), type="l", lwd=3)
#----------------------------------------------
### e)

cat("the expected value of the first method is:", mean(chain[-c(1:30)]), "\n")
cat("the expected value of the second method is:", mean(chain2[-c(1:30)]), "\n")
cat("the expected value of the third method is:", mean(chain3[-c(1:30)]), "\n")
#================================================================================
# 2.a
################################
### Boundary ellipse
### for Gibbs sampling example
### from Lecture 4
### Fall 2023, by Frank Miller
################################
w  <- 1.999
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4)  # a range of x1-values, where the term below the root is n
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)
# ellipse
lines(xv, -(w/2)*xv-sqrt(1-(1-w^2/4)*xv^2), lwd=2, col=8)
lines(xv, -(w/2)*xv+sqrt(1-(1-w^2/4)*xv^2), lwd=2, col=8)

# 2.c
#Gibbs sampling
f1_cond <- function(x2){
  return(runif(1, min = -0.9995*x2 - sqrt(1-0.00099975*x2), max = -0.9995*x2 + sqrt(1-0.00099975*x2)))
}
f2_cond <- function(x1){
  return(runif(1, min = -0.9995*x1 - sqrt(1-0.00099975*x1), max = -0.9995*x1 + sqrt(1-0.00099975*x1)))
}

set.seed(12345)
n <- 1000
x1 <- 0
x2 <- 0
X1 <- c()
X2 <- c()
for(i in 1:n){
  x1 <- f1_cond(x2)
  x2 <- f2_cond(x1)
  X1 <- c(X1, x1)
  X2 <- c(X2, x2)
}

w  <- 1.999
xv <- seq(-1, 1, by=0.01) * 1/sqrt(1-w^2/4)  # a range of x1-values, where the term below the root is n
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)
# ellipse
lines(xv, -(w/2)*xv-sqrt(1-(1-w^2/4)*xv^2), lwd=2, col=8)
lines(xv, -(w/2)*xv+sqrt(1-(1-w^2/4)*xv^2), lwd=2, col=8)
#Add the points X1 and X2 to the graph
points(X1, X2, col = "red", pch = 20, cex = 0.5)

#Determine P(X1 > 0)

means <- c()
for(i in 1:100){
  set.seed(i)
  n <- 1000
  x1 <- 0
  x2 <- 0
  X1 <- c()
  X2 <- c()
  for(i in 1:n){
    x1 <- f1_cond(x2)
    x2 <- f2_cond(x1)
    X1 <- c(X1, x1)
    X2 <- c(X2, x2)
  }
  means <- c(means, mean(X1 > 0))
}
#mean(means)

#2.e

#Gibbs sampling
xv <- seq(-sqrt(4000), sqrt(4000) , by=0.01)   # a range of x1-values, where the term below the root is
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1, ylim = c(-3,3))
# ellipse
lines(xv, -sqrt(1-xv^2*0.00025)/(0.99975), lwd=2, col=8)
lines(xv, sqrt(1-xv^2*0.00025)/(0.99975), lwd=2, col=8)

u1_given_ <- function(u2){
  return(runif( 1, min = -sqrt((1-u2^2*0.00025)/0.99975), max = sqrt((1-u2^2*0.00025)/0.99975) ))
}
u2_given_ <- function(u1){
  return(runif(1, min = -sqrt((1-u1^2*0.99975)/0.00025), max = sqrt((1-u1^2*0.99975)/0.00025) ))
}

means <- c()
for(i in 1:100){
```

```r
  set.seed(i)
  n <- 1000
  u1 <- 0
  u2 <- 0
  U1 <- c()
  U2 <- c()
  for(i in 1:n){
    u1 <- u1_given_(u2)
    u2 <- u2_given_(u1)
    U1 <- c(U1, u1)
    U2 <- c(U2, u2)
  }
  means <- c(means, mean(U1 > 0))
}
#mean(means)
points(U2, U1, col = "red", pch = 20, cex = 0.5)
```