732A90 Computational Statistics

# Lab 4

Johannes Hedström & Mikael Montén

STIMA
Institutionen för datavetenskap
Linköpings universitet

2024-01-25

## Contents

## 1 Statement of contribution

Mikael did assignment 2 and Johannes did assignment 1, we discussed our results afterwards.

## 2 Question 1: Computations with Metropolis–Hastings

Consider a random variable X with the following probability density function:

$$f(x) \propto x^5 e^{-x}, x > 0$$

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.

### 2.1 a

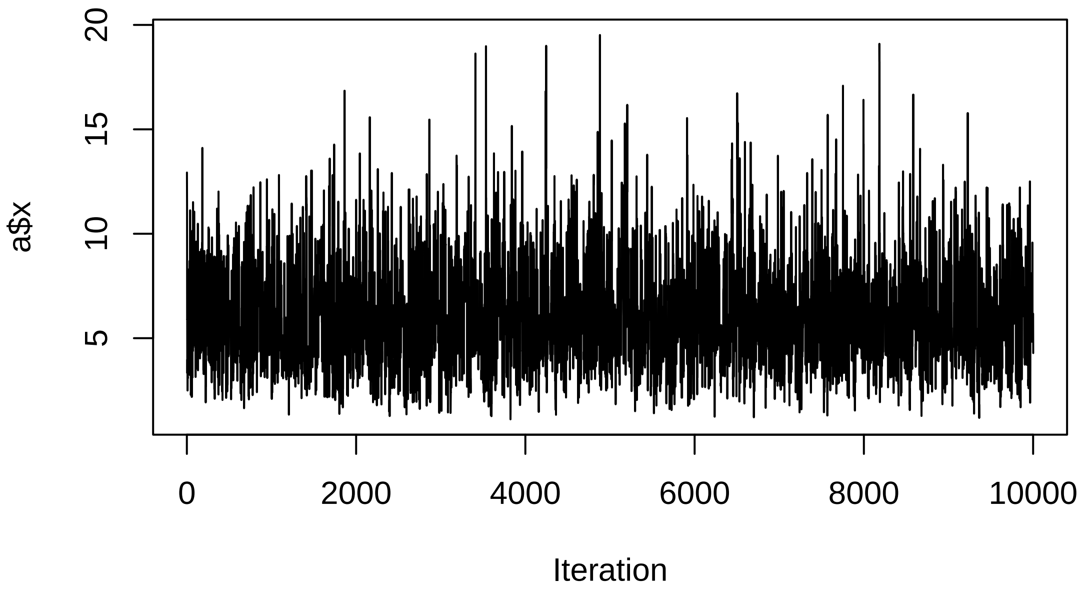
Use the Metropolis–Hastings algorithm to generate 10000 samples from this distribution by using a log–normal $LN(X_t, 1)$ proposal distribution; take some starting point. Plot the chain you obtained with iterations on the horizontal axis. What can you guess about the convergence of the chain? If there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

MH ratio

$$R(x^t, x^*) = \frac{f(x^*)g(x^t|x^*)}{f(x^t)g(x^*|x^t)}$$

```r
# number of samples
n <- 10000

# PDF
f <- function(x){
  if(all(x > 0)){
  x^5 * exp(-x)
  }
}



# log normal dist given x_star
g <- function(x,x_star,sd=1){

  dlnorm(x,log(x_star),sd=sd)
}
```

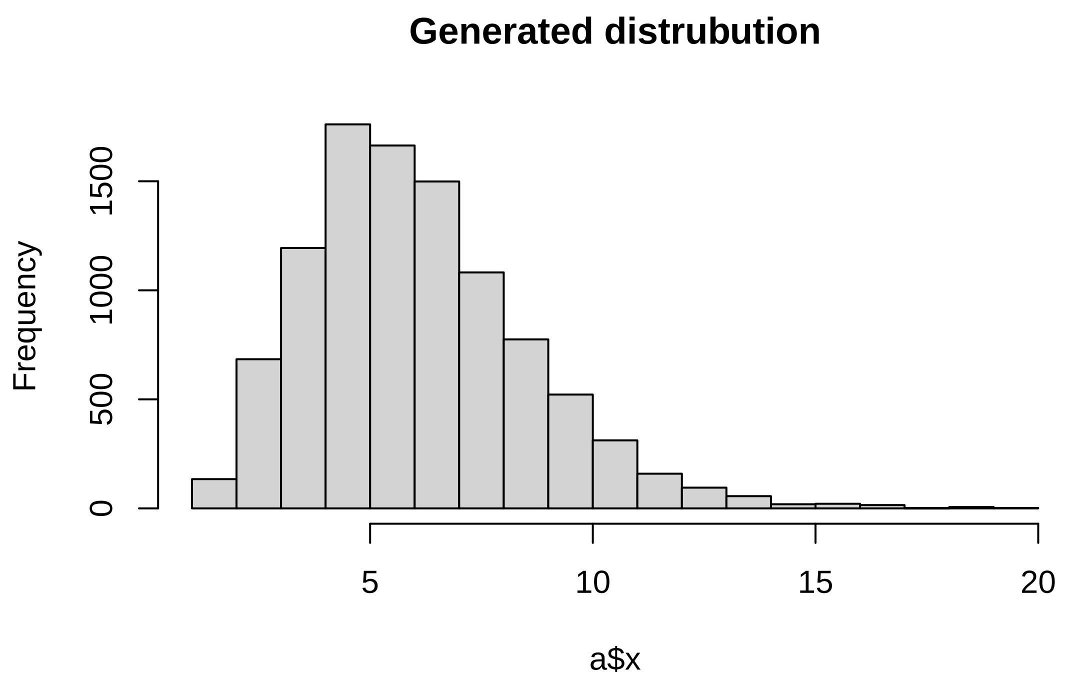
```r
# Metropolis-hasting
metro <- function(n){

  point <- runif(1,0.00001,15) # sampling starting point

  x <- c(point)
  reject <- 0
  for(i in 1:n){

    x_star <- rlnorm(1,log(x[i]),sdlog=1) # sample new point
    mhr <- (f(x_star) * g(x[i],x_star))/ (f(x[i]) * g(x_star,x[i])) # mhr ratio

    p <- runif(1,0,1) # generate prob
     if(mhr > p || mhr==1){ # accept step
     x <- c(x,x_star)
     }else{ # reject step
    x <- c(x,x[i])
    reject <- reject + 1
    }
  }
  return(list('x'=x,'rej'=reject))
}

a <- metro(10000)
plot(a$x, type='l', xlab='Iteration')
```



The burn-in period looks to be very small in the plot, looks to be under 100 iterations, actually looks less than that but as its hard to see a higher value than our initial thought was chosen.

We guess that the chain haven´t converged yet as the values are fluctuating with a high variance around the mean.

```r
hist(a$x, main='Generated distrubution')
```



```r
rej <- a$rej


knitr::kable(c('acceptence rate '= (n- rej)/n))
```

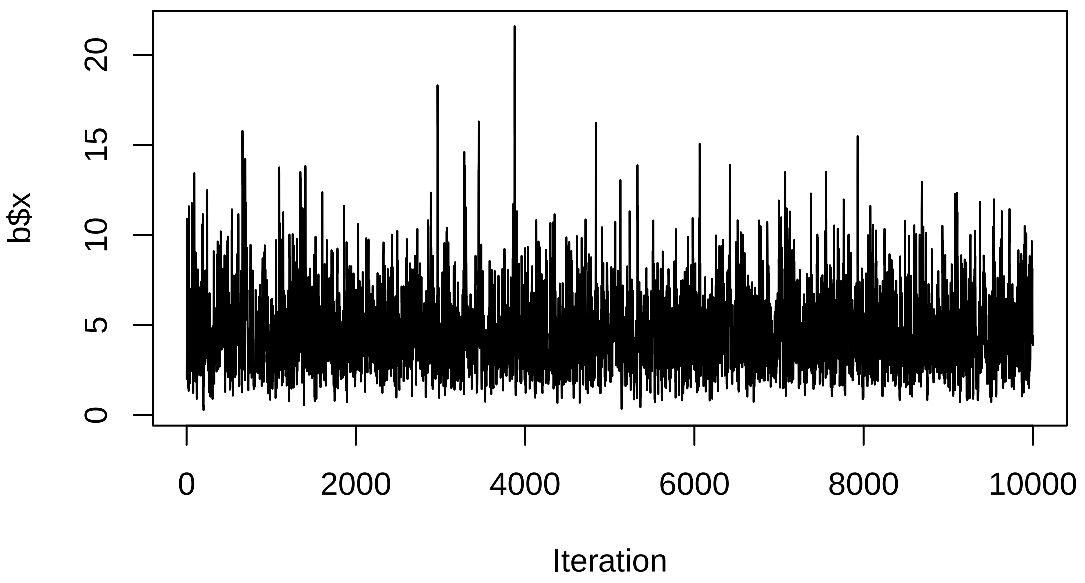
|  | x |
|---|---:|
| acceptence rate | 0.4442 |

The acceptance rate is around 44 percent.

## 2.2 b

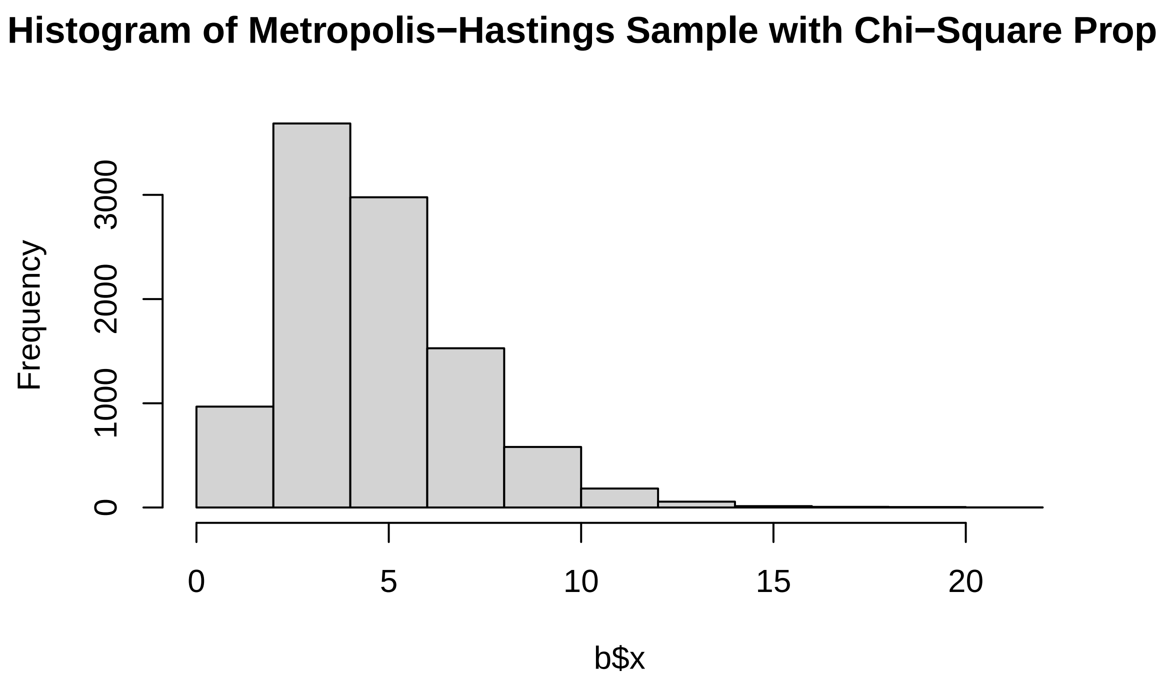
Perform Part a by using the chi–square distribution $\chi^2(\lfloor X_t + 1 \rfloor)$ as a proposal distribution, where $\lfloor x \rfloor$ is the floor function, meaning the integer part of x for positive x, i.e. $\lfloor 2.95 \rfloor = 2$

```r
g_chi_square <- function(x, x_star){
  # g function for chi dist
  dchisq(x_star, df = floor(x + 1))
}


# Metropolis-hasting with chi^2[X_t + 1] dist
metro_b <- function(n){

  point <- floor(runif(1,1,15)) # sampling starting point

  x <- c(point)
  reject <- 0
  for(i in 1:n){

    x_star <- rchisq(1,df = floor(x[i] + 1)) # sample new point from chi
    mhr <- (f(x_star) * g_chi_square(x[i],x_star))/ (f(x[i]) * g_chi_square(x_star,x[i])) # mhr ratio

    p <- runif(1,0,1) # generate prob
     if(mhr > p || mhr == 1){ # accept step
     x <- c(x,x_star)
     }else{ # reject step
    x <- c(x,x[i])
    reject <- reject + 1
    }
  }
  return(list('x'=x,'rej'=reject))
}

b<-metro_b(10000)
plot(b$x, type='l', xlab='Iteration')
```



Looks like about the same as above but maybe we generated a bit lower values, the burn in also looks to be quite small.

```
# Histogram of the sample
hist(b$x, main = "Histogram of Metropolis-Hastings Sample with Chi-Square Proposal")
```



We can here see that we have gotten lower values now, as the most values looks to be closer to 4 than 5-6 as it was in a, and the algorithm haven't converged yet.

```
rejb <- b$rej


knitr::kable(c('acceptence rate '= (n- rejb)/n))
```

| | x |
|---|---|
| acceptence rate | 0.603 |

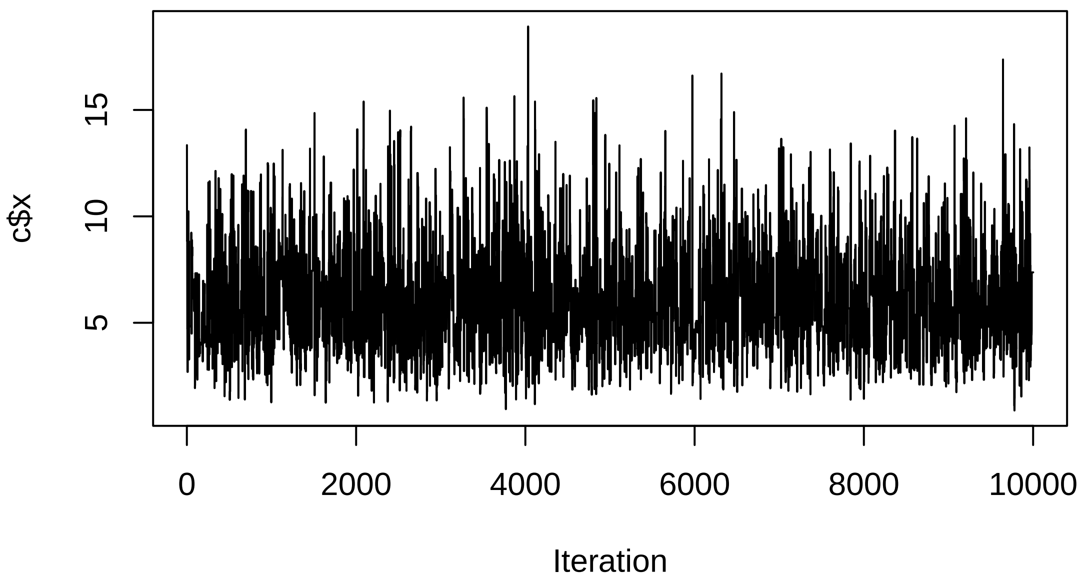
Now we accept more values then in a, close to 60 % of all $x^*$ values.

## 2.3 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

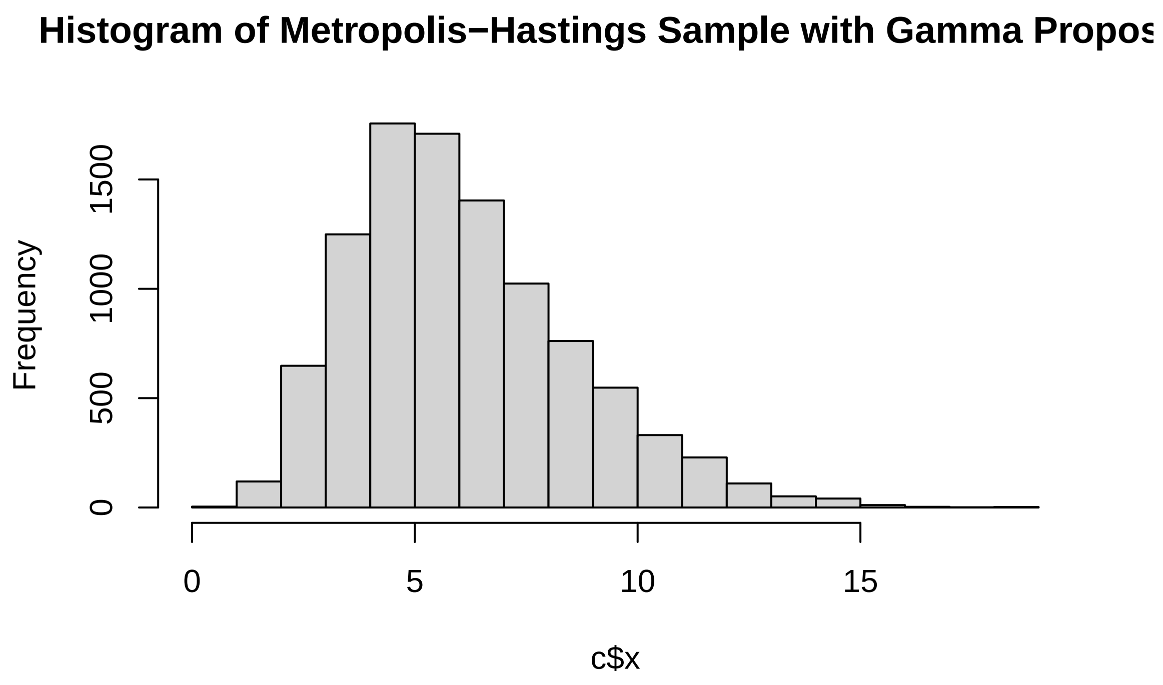
We are doing the log-normal $LN(X_t, 5)$ to see if it converges on the 10000 samples

```r
# Metropolis-hasting with chi^2[X_t + 1] dist
metro_c <- function(n){

  point <- runif(1,0.00001,15) # sampling starting point

  x <- c(point)
  reject <- 0
  for(i in 1:n){

    x_star <- rlnorm(1,log(x[i]),sdlog=log(5)) # sample new point
    mhr <- (f(x_star) * g(x[i],x_star,sd=log(5)))/ (f(x[i]) * g(x_star,x[i],sd=log(5))) # mhr ratio

    p <- runif(1,0,1) # generate prob
     if(mhr > p || mhr==1){ # accept step
     x <- c(x,x_star)
     }else{ # reject step
    x <- c(x,x[i])
    reject <- reject + 1
    }
  }
  return(list('x'=x,'rej'=reject))
}

c<-metro_c(10000)

plot(c$x, type='l', xlab='Iteration')
```



The algorithm haven't converged yet.

```r
# Histogram of the sample
hist(c$x, main = "Histogram of Metropolis-Hastings Sample with Gamma Proposal")
```



Here we got a histogram close to the one in task a, which is expected as its from the same distribution with another standard deviation.

```
rejc <- c$rej


knitr::kable(c('acceptence rate '= (n- rejc)/n))
```

| | x |
|---|---:|
| acceptence rate | 0.3014 |

The acceptance rate is now a bit lower than for a(~44%) and much lower than for b(~60%).

## 2.4 d

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

```
df <- data.frame(1-(rej/n),1-(rejb/n),1-(rejc/n))

colnames(df) <- c('a','b','c')

knitr::kable(df, caption='Acceptance rate for task a, b & c')
```

Table 4: Acceptance rate for task a, b & c

| a | b | c |
|---:|---:|---:|
| 0.4442 | 0.603 | 0.3014 |

The best acceptance rate for a one dimensional with normal target and proposal distribution problem accordingly to (Geof H. Givens 2012) is 44%. Our PDFs rent normal distributed but they are at least unimodal, so to be somewhere close to this rule should be a good acceptance rate and our log-normal $LN(X_t, 1)$ is very close to that.

The worst is the chi-square distribution which accepts around 60%, and this might be because they are too similar, it would have been better to take a distribution which have about the same tails as f(x) but differ from the higher density parts.

## 2.5 e

Estimate

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

using the samples from Parts a, b, and c

```r
df<-data.frame(mean(a$x),mean(b$x),mean(c$x))

colnames(df) <- c('E(X_a)', 'E(X_b)','E(X_c)')

knitr::kable(df,caption = 'E(X) for the 3 samples')
```

Table 5: E(X) for the 3 samples

| E(X_a) | E(X_b) | E(X_c) |
|---:|---:|---:|
| 6.06068 | 4.598308 | 6.090794 |

The E(X) is similar for the two samples with a proposal from the log-normal distribution, the chi-square sample is much lower.

## 2.6 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.

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx$$

But the mean for $Gamma(\alpha, \lambda)$ is $\frac{\alpha}{\lambda}$ which in our case is $\frac{6}{1} = 6$ (“Gamma Distribution” 2023). So our samples with the log-normal distribution are close to the true mean of the distribution.

## 3 Question 2: Gibbs sampling

Let $X = (X_1, X_2)$ be a bivariate distribution with density $f(x_1, x_2) \propto 1\{x_1^2 + wx_1x_2 + x_2^2 < 1\}$ for some specific $w$ with $|w| < 2$. $X$ has a uniform distribution on some two-dimensional region. WE consider here the case $w = 1.999$.

### 3.1 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.*

```
set.seed(12345)

# solve the density function with w = 1.999 and create lower + upper bounds
upper <- function(x){
  ((sqrt(4000000 - 3999 * x^2)) - 1999 * x) / 2000
}

lower <- function(x){
  (-(sqrt(4000000 - 3999 * x^2)) - 1999 * x) / 2000
}

# range of x1 values, ensuring the term below the square root is defined at all values
xv <- seq(-sqrt(4000000/3999)+0.01, sqrt(4000000/3999)-0.01, by = 0.01)

# create upper and lower boundaries
u_bd <- upper(xv)
l_bd <- lower(xv)
```

```
# plot
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)
lines(xv, l_bd)
lines(xv, u_bd)
```

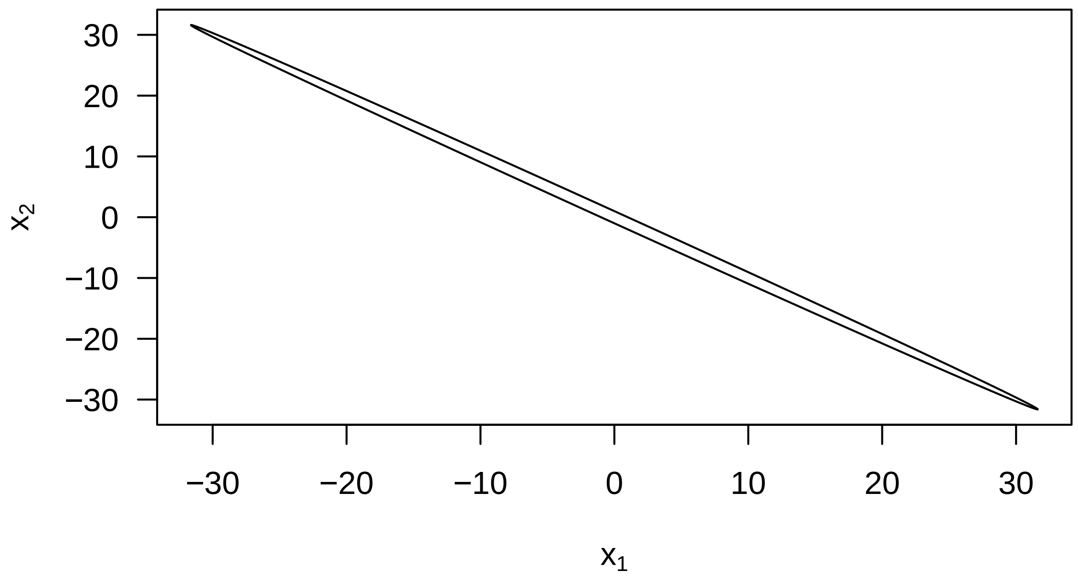


Figure 1: Distribution boundaries where X is uniform

The upper and lower bound is calculated by solving $x_1^2 + 1.999 \cdot x_1 x_2 + x_2^2 - 1 = 0$ with respect to $x_2$. However, solving for $x_1$ yields the same result as the density function is similar for both possible conditional distributions.

The term under the square root is always defined where $4000000 - 3999x^2 > 0 <=> 3999x^2 \leq 4000000 <=> x^2 \leq 4000000/3999 <=> x \leq \sqrt{400000/3999}$.

## 3.2 b.

*What is the conditional distribution of $X_1$ given $X_2$ and that of $X_2$ given $X_1$?*

The conditional distribution for $X_1$ given $X_2$ is

$$\left(\frac{-\sqrt{4000000 - 3999 \cdot x_2^2} - 1999 \cdot x_2}{2000}, \frac{\sqrt{4000000 - 3999 \cdot x_2^2} - 1999 \cdot x_2}{2000}\right)$$

The distribution is the same for both conditions, just replace $x_2$ with $x_1$, alas the conditional distribution for $X_2$ given $X_1$ is

$$\left(\frac{-\sqrt{4000000 - 3999 \cdot x_1^2} - 1999 \cdot x_1}{2000}, \frac{\sqrt{4000000 - 3999 \cdot x_1^2} - 1999 \cdot x_1}{2000}\right)$$

## 3.3 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 $P(X_1 > 0)$ 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?

```r
# conditional distribution of X1 given X2
cond_x1 <- function(x2) {
  a <- (-(sqrt(4000000 - 3999 * x2^2)) - 1999 * x2) / 2000
  b <- ((sqrt(4000000 - 3999 * x2^2)) - 1999 * x2) / 2000
  runif(1, min = a, max = b)
}

# conditional distribution of X2 given X1
cond_x2 <- function(x1) {
  a <- (-(sqrt(4000000 - 3999 * x1^2)) - 1999 * x1) / 2000
  b <- ((sqrt(4000000 - 3999 * x1^2)) - 1999 * x1) / 2000
  runif(1, min = a, max = b)
}

# gibbs
gibbs <- function(n, x1, x2){
  samples <-matrix(0,nrow=n, ncol=2)

  for (i in 1:n) {
    x1 <- cond_x1(x2)
    x2 <- cond_x2(x1)
    samples[i,] <- c(x1, x2)
  }

  return(samples)
}
```

```
set.seed(12345)
# plot from a)
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)
lines(xv, l_bd)
lines(xv, u_bd)
# gibbs sample
gibbs_sample <- gibbs(1000,1,1) # starting value 1,1
points(gibbs_sample)
```

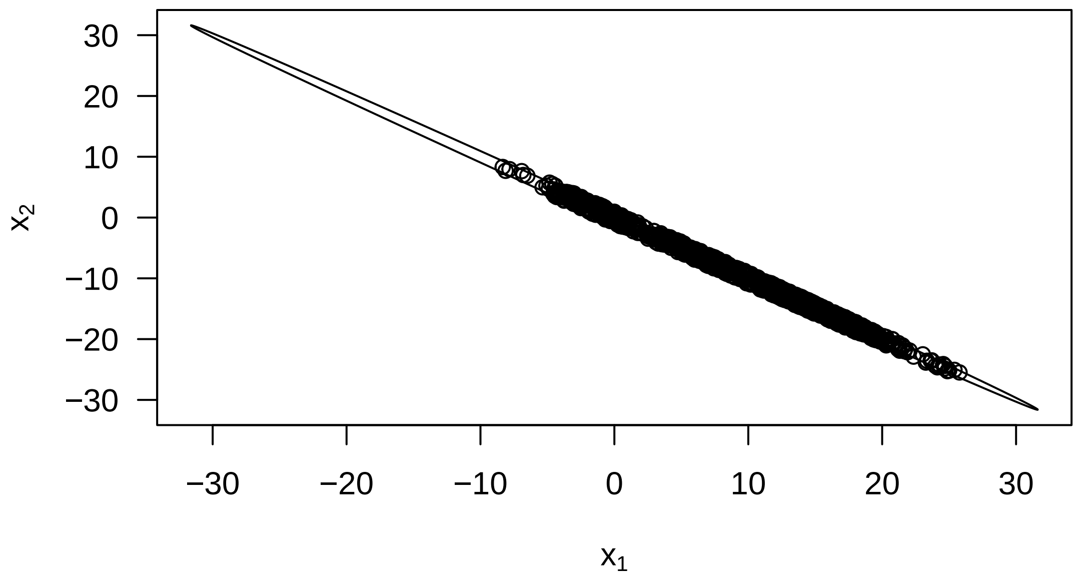


Figure 2: Sampled points within the boundaries

The whole space for where the density is uniform is not used, it's rather around half of it. Here, the starting points for the algorithm is $(1, 1)$ and other starting values might affect this, but in general there is a case to be made for the algorithm not sampling the entire space very well.

```
# First sample gives us 881/1000 above 1 so P(X1 > 0) = 85.3%
gs1 <- table(gibbs_sample[,1] > 0)

# Second sample
gibbs_sample1 <- gibbs(1000,1,1)
gs2 <- table(gibbs_sample1[,1] > 0)
# P(X1 > 0) = 41.5%
```

```r
# Thirds sample
gibbs_sample2 <- gibbs(1000,1,1)
gs3 <- table(gibbs_sample2[,1] > 0)
# P(X1 > 0) = 17.3%

gs_comparison <- rbind(gs1,gs2,gs3)
rownames(gs_comparison) <- c("Sample 1", "Sample 2", "Sample 3")
knitr::kable(gs_comparison, caption = "Probabilities for X1 > 0")
```

Table 6: Probabilities for X1 > 0

| | FALSE | TRUE |
|---|---|---|
| Sample 1 | 147 | 853 |
| Sample 2 | 585 | 415 |
| Sample 3 | 827 | 173 |

The 3 different samplings result in three very different results for $P(X_1 > 0)$. This indicates that the true result for the probability in question does not have a clear answer, but if we were to average the three samples we got we would get $\frac{0.853+0.415+0.173}{3} = 0.48$ which might be a better point estimation than each single one and looks more reasonable by analyzing the ellipsoid. This can of course be more extensively researched with more samples. This also shows that the sampling doesn't work very well for each case as the point estimation differs so much, meaning there is a concentration of points where the algorithm samples from instead of the whole space.

## 3.4 d.

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

The increase in $w$ has made the uniform area a lot more narrow than it was in the lecture. This means the conditional distribution of each variable is more narrow which makes it hard for the algorithm to sample distribution well. The resulting ellipsoid when increasing $w$ also has a showcases a higher correlation between the variables which seems to hinder Gibbs sampling to converge quick, which can be avoided using blocking (Geof H. Givens 2012).

## 3.5 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 oin. 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 resultrs with Part c.

```
upper_u2 <- function(u){
  sqrt(4000-u^2) / sqrt(3999)
}

lower_u2 <- function(u){
  -(sqrt(4000-u^2) / sqrt(3999))
}

upper_u1 <- function(u){
  sqrt(4000-3999*u^2)
}

lower_u1 <- function(u){
  -(sqrt(4000-3999*u^2))
}
```

By solving $x_1^2 + w \cdot x_1 x_2 + x_2^2 - 1 = 0$ and replacing $x_1$ with $(u_2 + u_1)/2$ and $x_2$ with $(u_2 - u_1)/2$ we acquire new boundaries for the density. The new boundaries are

$$u_2 = (-\sqrt{4000 - u^2}/\sqrt{3999}, \sqrt{4000 - u^2}/\sqrt{3999})$$

$$u_1 = (-\sqrt{4000 - 3999 \cdot u^2}, \sqrt{4000 - 3999 \cdot u^2})$$

.

The $u_2$ boundaries are defined when the square root values are non-negative and the denominator must be $\neq 0$. This means the boundaries for $u_2$ always are defined at $4000 - u^2 \geq 0 <=> u^2 \leq 4000 <=> -\sqrt{4000} < u < \sqrt{4000}$ since $\sqrt{3999} \neq 0$.

For $u_1$, $4000 - 3999 \cdot u^2 \geq 0 <=> 3999 \cdot u^2 \leq 4000 <=> u^2 \leq 4000/3999 <=> -\sqrt{4000/3999} \leq u \leq \sqrt{4000/3999}$.

```r
xv_u2 <- seq(-sqrt(4000), sqrt(4000), by = 0.01)

# create upper and lower boundaries
u_bd_u <- upper_u2(xv_u2)
l_bd_u <- lower_u2(xv_u2)

# plot w.r.t to u2, and scale the plots yaxis with the u1 limits
plot(xv_u2, xv_u2, type="n", xlab=expression(u[1]), ylab=expression(u[2]),
     las=1, ylim=c(-sqrt(4000/3999), sqrt(4000/3999)))
lines(xv_u2, l_bd_u)
lines(xv_u2, u_bd_u)
```

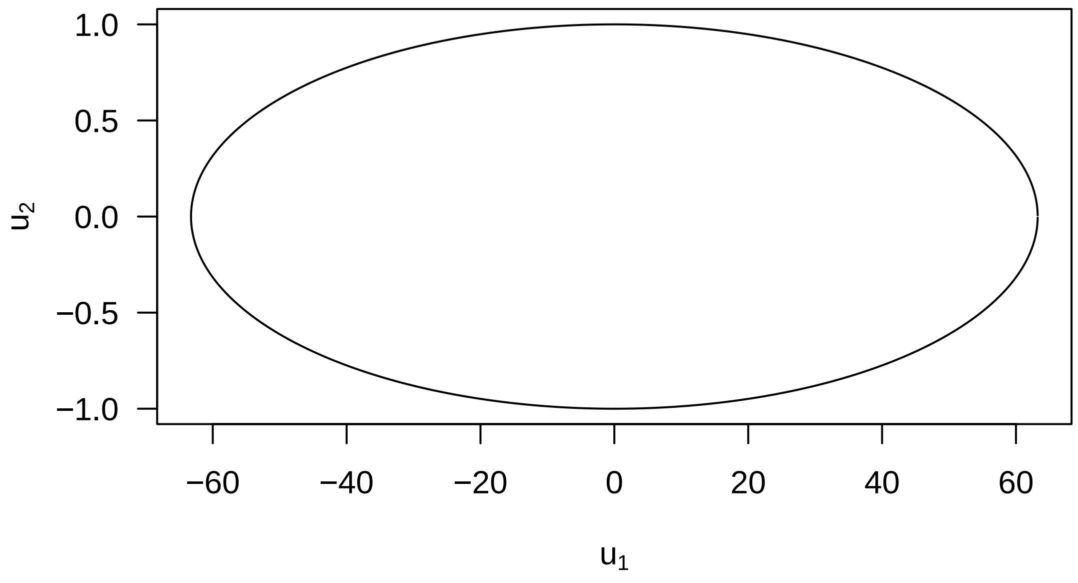


Figure 3: Decision boundaries where the transformed U variables are uniform

The result from transforming to $U$ is a decision boundary that no longer is at an angle, and instead is uniform across a larger surface area of it's defined limits. However, the transformation has made the $u_1$ boundary have a greater span but the $u_2$ boundary has a much lower span.

```r
# conditional distribution of U1 given U2
cond_u1 <- function(u2) {
  a <- -(sqrt(4000-3999*u2^2))
  b <- (sqrt(4000-3999*u2^2))
  runif(1, min = a, max = b)
}

# conditional distribution of U2 given U1
cond_u2 <- function(u1) {
  a <- -(sqrt(4000-u1^2) / sqrt(3999))
  b <- sqrt(4000-u1^2) / sqrt(3999)
  runif(1, min = a, max = b)
}


# gibbs
gibbs_u <- function(n, u1, u2){
  samples <-matrix(0,nrow=n, ncol=2)

  for (i in 1:n) {
    u1 <- cond_u1(u2)
    u2 <- cond_u2(u1)
    samples[i,] <- c(u1, u2)
  }

  return(samples)
}
```

```r
gibbs_u_sample <- gibbs_u(1000,1,1)
plot(xv_u2, xv_u2, type="n", xlab=expression(u[1]), ylab=expression(u[2]), las=1, ylim=c(-1, 64/63))
lines(xv_u2, l_bd_u)
lines(xv_u2, u_bd_u)
points(gibbs_u_sample)
```

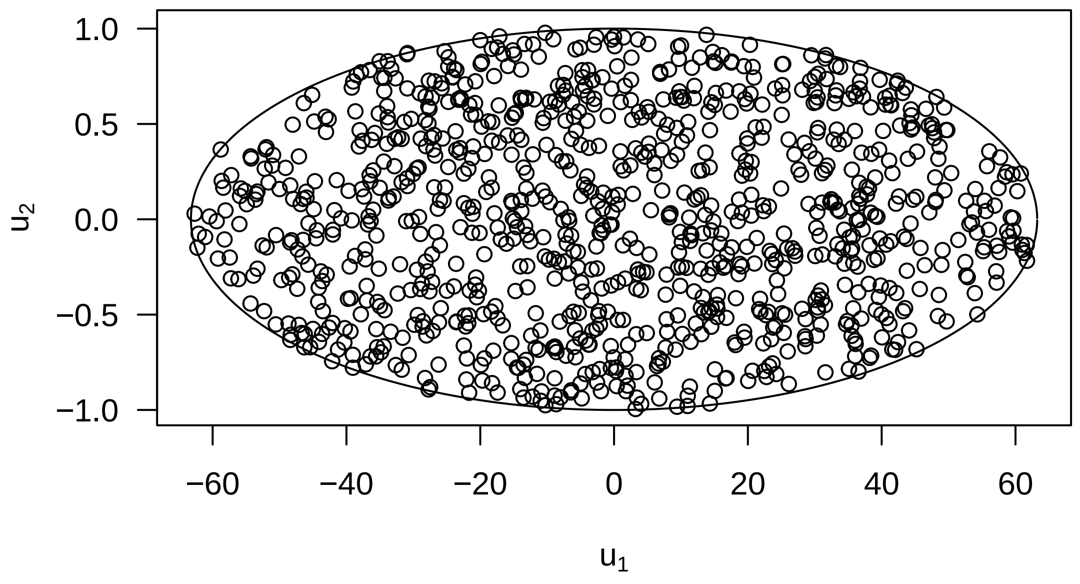


Figure 4: Decision boundary with sampled points plotted

The transformation resulted in an algorithm that uses the whole defined space for where the algorithm is uniform and therefore works better than the previous one.

```r
gs_u <- table((gibbs_u_sample[,1] + gibbs_u_sample[,2])/2 > 0)
knitr::kable(gs_u, caption = "Probability of X1 > 0 for U1 and U2")
```

Table 7: Probability of X1 > 0 for U1 and U2

| Var1 | Freq |
|---|---|
| FALSE | 509 |
| TRUE | 491 |

The result is no more in lined with what you would expect when you analyze the decision boundary ellipsoid visually. If we were to draw several of these samples and took an average of them it would rather quick converge to a very good point estimator.

## 4 References

"Gamma Distribution." 2023. Wikipedia. https://en.wikipedia.org/wiki/Gamma_distribution.

Geof H. Givens, Jennifer A.Hoeting. 2012. *Computational Statistics*. 2nd ed. Wiley. https://doi.org/https://doi.org/10.1002/9781118555552.

## 5 Appendix

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning=FALSE, fig.width = 6, fig.height = 4)
# number of samples
n <- 10000

# PDF
f <- function(x){
  if(all(x > 0)){
  x^5 * exp(-x)
  }
}


# log normal dist given x_star
g <- function(x,x_star,sd=1){

  dlnorm(x,log(x_star),sd=sd)
}




# Metropolis-hasting
metro <- function(n){

  point <- runif(1,0.00001,15) # sampling starting point

  x <- c(point)
  reject <- 0
  for(i in 1:n){

    x_star <- rlnorm(1,log(x[i]),sdlog=1) # sample new point
    mhr <- (f(x_star) * g(x[i],x_star))/ (f(x[i]) * g(x_star,x[i])) # mhr ratio

    p <- runif(1,0,1) # generate prob
     if(mhr > p || mhr==1){ # accept step
     x <- c(x,x_star)
     }else{ # reject step
    x <- c(x,x[i])
    reject <- reject + 1
    }
```

```r
  }
  return(list('x'=x,'rej'=reject))
}

a <- metro(10000)
plot(a$x, type='l', xlab='Iteration')

hist(a$x, main='Generated distrubution')
rej <- a$rej


knitr::kable(c('acceptence rate '= (n- rej)/n))
g_chi_square <- function(x, x_star){
  # g function for chi dist
  dchisq(x_star, df = floor(x + 1))
}


# Metropolis-hasting with chi^2[X_t + 1] dist
metro_b <- function(n){

  point <- floor(runif(1,1,15)) # sampling starting point

  x <- c(point)
  reject <- 0
  for(i in 1:n){

    x_star <- rchisq(1,df = floor(x[i] + 1)) # sample new point from chi
    mhr <- (f(x_star) * g_chi_square(x[i],x_star))/ (f(x[i]) * g_chi_square(x_star,x[i])) # mhr ratio

    p <- runif(1,0,1) # generate prob
     if(mhr > p || mhr == 1){ # accept step
     x <- c(x,x_star)
     }else{ # reject step
    x <- c(x,x[i])
    reject <- reject + 1
    }
  }
  return(list('x'=x,'rej'=reject))
}

b<-metro_b(10000)
plot(b$x, type='l', xlab='Iteration')
# Histogram of the sample
hist(b$x, main = "Histogram of Metropolis-Hastings Sample with Chi-Square Proposal")


rejb <- b$rej
```

```r
knitr::kable(c('acceptence rate '= (n- rejb)/n))

# Metropolis-hasting with chi^2[X_t + 1] dist
metro_c <- function(n){

  point <- runif(1,0.00001,15) # sampling starting point

  x <- c(point)
  reject <- 0
  for(i in 1:n){

    x_star <- rlnorm(1,log(x[i]),sdlog=log(5)) # sample new point
    mhr <- (f(x_star) * g(x[i],x_star,sd=log(5)))/ (f(x[i]) * g(x_star,x[i],sd=log(5))) # mhr ratio

    p <- runif(1,0,1) # generate prob
     if(mhr > p || mhr==1){ # accept step
     x <- c(x,x_star)
     }else{ # reject step
    x <- c(x,x[i])
    reject <- reject + 1
    }
  }
  return(list('x'=x,'rej'=reject))
}

c<-metro_c(10000)

plot(c$x, type='l', xlab='Iteration')



# Histogram of the sample
hist(c$x, main = "Histogram of Metropolis-Hastings Sample with Gamma Proposal")


rejc <- c$rej


knitr::kable(c('acceptence rate '= (n- rejc)/n))
df <- data.frame(1-(rej/n),1-(rejb/n),1-(rejc/n))

colnames(df) <- c('a','b','c')

knitr::kable(df, caption='Acceptance rate for task a, b & c')

df<-data.frame(mean(a$x),mean(b$x),mean(c$x))

colnames(df) <- c('E(X_a)', 'E(X_b)','E(X_c)')
```

```r
knitr::kable(df,caption = 'E(X) for the 3 samples')



set.seed(12345)

# solve the density function with w = 1.999 and create lower + upper bounds
upper <- function(x){
  ((sqrt(4000000 - 3999 * x^2)) - 1999 * x) / 2000
}

lower <- function(x){
  (-(sqrt(4000000 - 3999 * x^2)) - 1999 * x) / 2000
}

# range of x1 values, ensuring the term below the square root is defined at all values
xv <- seq(-sqrt(4000000/3999)+0.01, sqrt(4000000/3999)-0.01, by = 0.01)

# create upper and lower boundaries
u_bd <- upper(xv)
l_bd <- lower(xv)
# plot
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)
lines(xv, l_bd)
lines(xv, u_bd)
# conditional distribution of X1 given X2
cond_x1 <- function(x2) {
  a <- (-(sqrt(4000000 - 3999 * x2^2)) - 1999 * x2) / 2000
  b <- ((sqrt(4000000 - 3999 * x2^2)) - 1999 * x2) / 2000
  runif(1, min = a, max = b)
}

# conditional distribution of X2 given X1
cond_x2 <- function(x1) {
  a <- (-(sqrt(4000000 - 3999 * x1^2)) - 1999 * x1) / 2000
  b <- ((sqrt(4000000 - 3999 * x1^2)) - 1999 * x1) / 2000
  runif(1, min = a, max = b)
}

# gibbs
gibbs <- function(n, x1, x2){
  samples <-matrix(0,nrow=n, ncol=2)

  for (i in 1:n) {
    x1 <- cond_x1(x2)
    x2 <- cond_x2(x1)
    samples[i,] <- c(x1, x2)
  }
  return(samples)
}
set.seed(12345)
# plot from a)
plot(xv, xv, type="n", xlab=expression(x[1]), ylab=expression(x[2]), las=1)
lines(xv, l_bd)
lines(xv, u_bd)
# gibbs sample
gibbs_sample <- gibbs(1000,1,1) # starting value 1,1
points(gibbs_sample)
# First sample gives us 881/1000 above 1 so P(X1 > 0) = 85.3%
gs1 <- table(gibbs_sample[,1] > 0)

# Second sample
gibbs_sample1 <- gibbs(1000,1,1)
gs2 <- table(gibbs_sample1[,1] > 0)
# P(X1 > 0) = 41.5%

# Thirds sample
gibbs_sample2 <- gibbs(1000,1,1)
gs3 <- table(gibbs_sample2[,1] > 0)
# P(X1 > 0) = 17.3%

gs_comparison <- rbind(gs1,gs2,gs3)
rownames(gs_comparison) <- c("Sample 1", "Sample 2", "Sample 3")
knitr::kable(gs_comparison, caption = "Probabilities for X1 > 0")
upper_u2 <- function(u){
  sqrt(4000-u^2) / sqrt(3999)
}

lower_u2 <- function(u){
  -(sqrt(4000-u^2) / sqrt(3999))
}

upper_u1 <- function(u){
  sqrt(4000-3999*u^2)
}

lower_u1 <- function(u){
  -(sqrt(4000-3999*u^2))
}
xv_u2 <- seq(-sqrt(4000), sqrt(4000), by = 0.01)

# create upper and lower boundaries
u_bd_u <- upper_u2(xv_u2)
l_bd_u <- lower_u2(xv_u2)

# plot w.r.t to u2, and scale the plots yaxis with the u1 limits
```

```r
plot(xv_u2, xv_u2, type="n", xlab=expression(u[1]), ylab=expression(u[2]),
     las=1, ylim=c(-sqrt(4000/3999), sqrt(4000/3999)))
lines(xv_u2, l_bd_u)
lines(xv_u2, u_bd_u)
# conditional distribution of U1 given U2
cond_u1 <- function(u2) {
  a <- -(sqrt(4000-3999*u2^2))
  b <- (sqrt(4000-3999*u2^2))
  runif(1, min = a, max = b)
}

# conditional distribution of U2 given U1
cond_u2 <- function(u1) {
  a <- -(sqrt(4000-u1^2) / sqrt(3999))
  b <- sqrt(4000-u1^2) / sqrt(3999)
  runif(1, min = a, max = b)
}


# gibbs
gibbs_u <- function(n, u1, u2){
  samples <-matrix(0,nrow=n, ncol=2)

  for (i in 1:n) {
    u1 <- cond_u1(u2)
    u2 <- cond_u2(u1)
    samples[i,] <- c(u1, u2)
  }

  return(samples)
}
gibbs_u_sample <- gibbs_u(1000,1,1)
plot(xv_u2, xv_u2, type="n", xlab=expression(u[1]), ylab=expression(u[2]), las=1, ylim=c(-1, 64/63))
lines(xv_u2, l_bd_u)
lines(xv_u2, u_bd_u)
points(gibbs_u_sample)

gs_u <- table((gibbs_u_sample[,1] + gibbs_u_sample[,2])/2 > 0)
knitr::kable(gs_u, caption = "Probability of X1 > 0 for U1 and U2")
```