Convergence Rate Bounds for Iterative Random Functions Using One-Shot Coupling

Sabrina Sixta

08/10/2021

An example of a non linear autoregressive process

```
# D^2 = - inf funcMin(x,y)
f <- function(x,y){return( 0.25*(y-x-sin(y)+sin(x)) )}
g <- function(x,y){return( 0.5 *(x-y+sin(y)-sin(x)) )}
h <- function(x,y){return( 0.25*(y+x-sin(y)-sin(x)) )}

num <- function(x,y){
  num <-g(x,y)^2 + 4*exp(-0.5)*g(x,y)*sin(f(x,y))*cos(h(x,y))+2*sin(f(x,y))^2*(1+exp(-2)*(cos(h(x,y))^2
  return( 0.5*sqrt(num) )}

funcMin <- function(par){
  x <- par[1]
  y <- par[2]
  return(-abs(num(x,y))/abs(x-y))
}
```

The following provides an estimate of D when the optim function is used.

```
D <- optim(par = c(0.1,0.2), funcMin)
-D$value
```

```
## [1] 0.6693833
```

The following is a graph of funcMin to show that the estimated minimum is correct.

```
funcMin <- function(x, y){
  return(-abs(num(x,y))/abs(x-y))
}

x <- seq(-100, 100, length= 1000)
y <- x
z <- outer(x, y, funcMin)
```

```
fig <- plot_ly(x = x, y = y, z = z) %>% add_surface()

# fig
```

Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $X_0 = 1$ and $X_0' = 2$

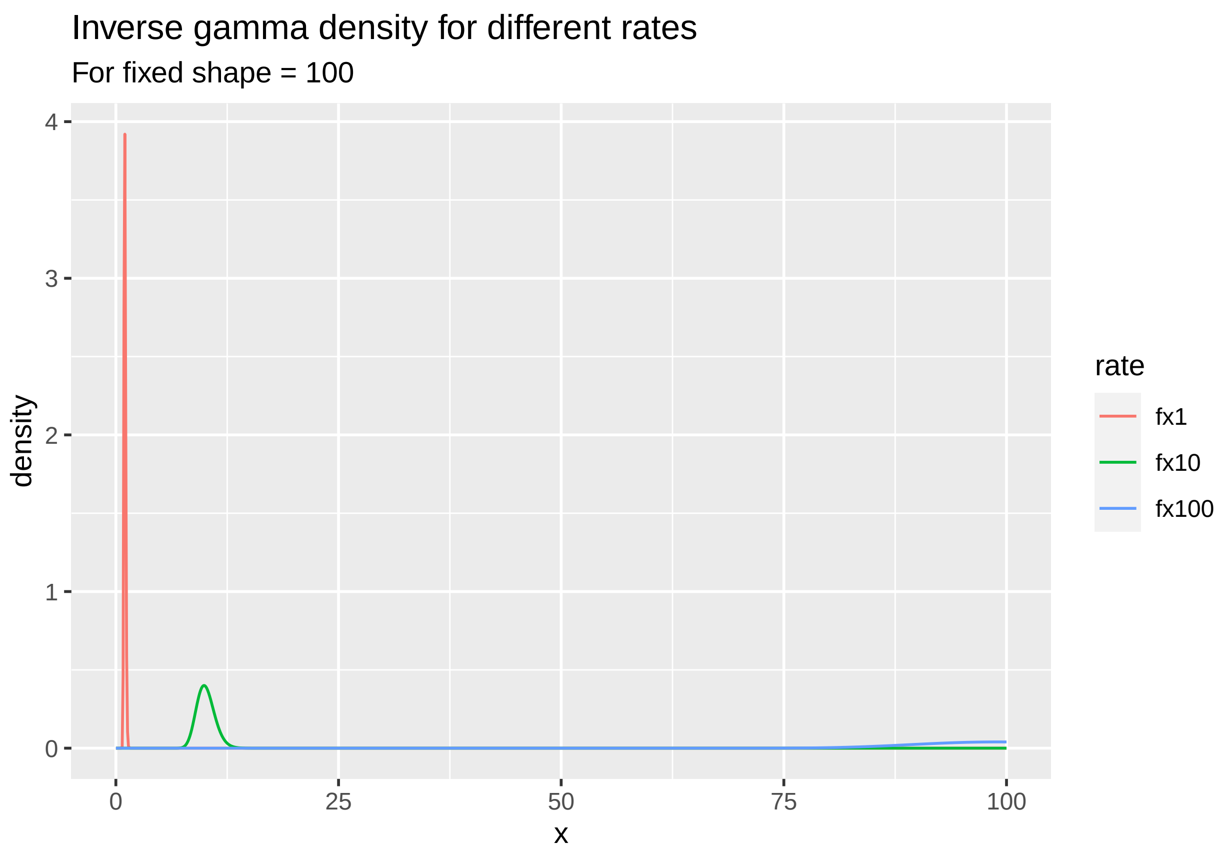
```
n <- 2*log(0.01 * sqrt(3*pi/2))/log(-D$value[1])
ceiling(n)+1
```

```
## [1] 21
```

Bayesian regression Gibbs sampler example

Code to generate figure 5.

```
library(invgamma)

x <- seq(0.01, 100, length=1000)
fx1 <- dinvgamma(x, shape=100, rate=100)
fx10 <- dinvgamma(x, shape=100, rate=1000)
fx100 <- dinvgamma(x, shape=100, rate=10000)
df <- data.frame(x, fx1, fx10, fx100)

df %>%
  select(x, fx1, fx10, fx100) %>%
  pivot_longer(cols = starts_with("fx"), names_to = "rate", values_to = "density") %>%
  ggplot(aes(x=x, y=density, col=rate)) +
  geom_line()+
  labs(title="Inverse gamma density for different rates", subtitle = "For fixed shape = 100")
```



Numerical example

```
dataPHD <- read.csv2(file="phd-delays.csv")
colnames(dataPHD) <- c("diff", "child", "sex","age","age2")
```

```
mDataPHD <- as.matrix(dataPHD)
Y <- mDataPHD[,1]
X <- mDataPHD[,-1]
```

```
#set parameters
lambda <- 2

#get variables
k <- nrow(X)
p <- ncol(X)
A <- t(X) %*% X + lambda*diag(p)
C <- t(Y) %*% (diag(k)-X %*% inv(A) %*% t(X)) %*% Y
C <- C[1,1]
K <- ((k/2+p+1)/exp(1))^(k/2+p)/gamma(k/2+p)*(k+2*p+2)/(C*exp(1))
D <- p/(k+p-2)
```

Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $X_0 = 1$ and $X_0' = 1001$

```
n <- log(0.01/(K*1000))/log(D)
ceiling(n)+1
```

```
## [1] 3
```

Bayesian location model Gibbs sampler

Numerical example Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $X_0 = 1$ and $X_0' = 2$.

```
data(trees)
df <- trees$Girth

j <- length(df)
S <- sum((df-mean(df))^2)
C <- ((j+1)/2)^((j-1)/2)*exp(-((j+1)/2))/((j+1)*gamma((j-1)/2))*S
#K <- (S/2)^((j-1)/2) * ((j+1)/S)^((j-3)/2) * exp(-(j+1)/2) / gamma((j-1)/2)
C
```

```
## [1] 13.74027
```

```
D <- 1/j
n <- log(0.01/(C*1000))/log(D)
ceiling(n) + 1
```

```
## [1] 6
```

Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $X_0 = 1$ and $X_0' = 2$ using the method from the one-shot coupling paper.

```
C <- j/2 +1
C
```

```
## [1] 16.5
```

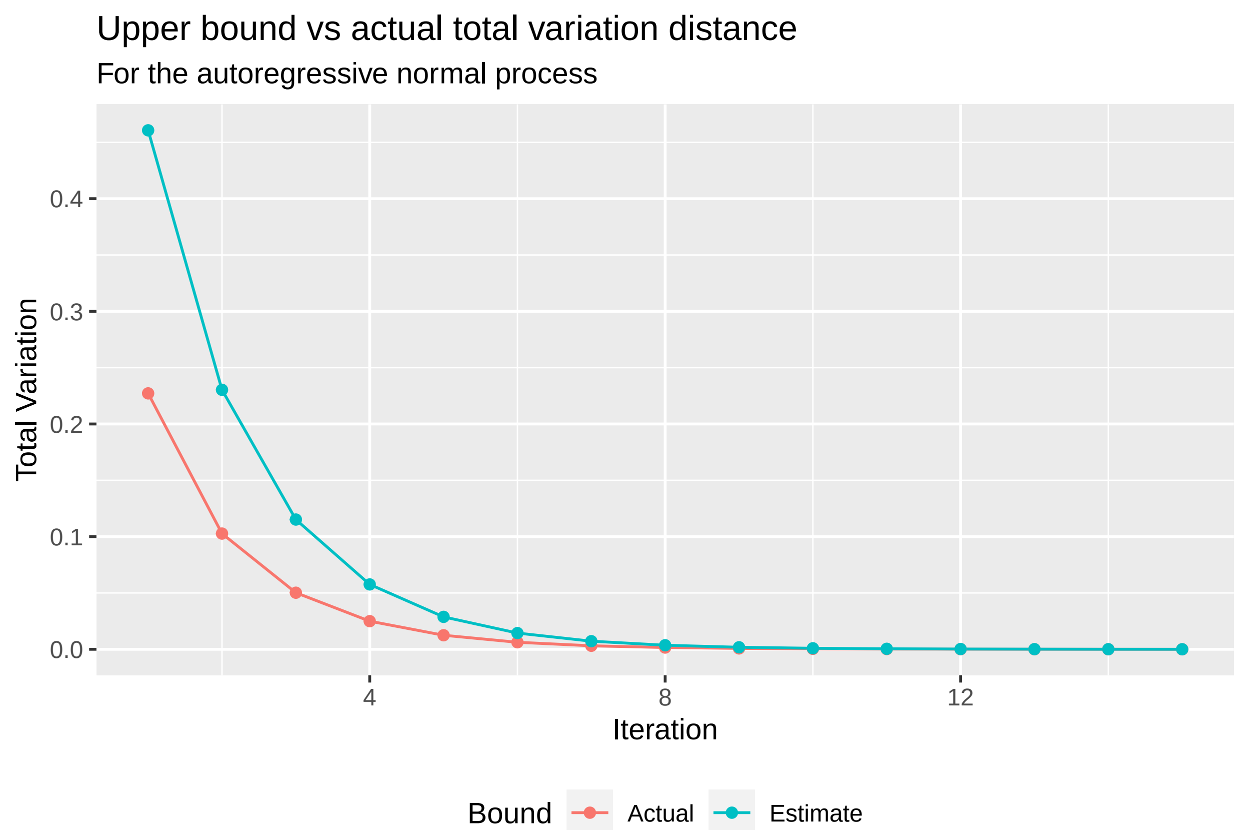
```
n <- log(0.01/(C*1000))/log(D)
ceiling(n)
```

```
## [1] 5
```

Autoregressive normal process

Code to generate figure 1.

```r
x0=1
y0=0
n = 15
actualBound = 1:n; upperBound = 1:n
for(i in 1:n){
  actualBound[i] <- 1-2*pnorm(-abs(x0-y0)/(2*sqrt(4^i-1)))
  #actualBound[i] <- 1-2*pnorm(-2^(-i-1)*abs(x0-y0)*(1-4^(-i))^(-0.5))
  upperBound[i] <- sqrt(2/(3*pi))*abs(x0-y0)/2^(i-1)
}
df <- data.frame(Iteration=1:n, Actual=actualBound, Estimate=upperBound)

df %>%
  pivot_longer(cols = Actual:Estimate, names_to = "Bound", values_to = "value") %>%
  ggplot(aes(x=Iteration, y=value, col=Bound)) +
  geom_line()+
  geom_point() +
  labs(fill = "Bound", y="Total Variation", title = "Upper bound vs actual total variation distance", su
  theme(legend.position = "bottom")
```



Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $x_0 = 0$ and $x_0' = 1$.

```r
n <- log(sqrt(3*pi/2)*0.01)/log(0.5)
ceiling(n) +1
```

```
## [1] 7
```

AR normal process with d independent coordinates

Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $x_0 = \vec{0}_{100}$ and $x'_0 = \vec{1}_{100}$.

```
d <- 100
n <- log(0.01/(d*sqrt(2/(3*pi))))/log(0.5)
ceiling(n)+1
```

```
## [1] 14
```

AR normal process with d dependent coordinates

```
d <- 100
#diag(0.5, nrow=10)

X <- rep(1, d)
Y <- rep(0, d)
m <- matrix(0,d,d)
diag(m[-1,])<-0.125
diag(m[,-1])<-0.125
diag(m)<-0.5
sigma <- m
A <- m

eigenM <- eigen(m)
#D <- diag(eigenM$values)
P <- eigenM$vectors
```

Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $x_0 = \vec{0}_{100}$ and $x'_0 = \vec{1}_{100}$.

```
C <- sqrt(d)/sqrt(2*pi) * norm(inv(sigma), "F") * norm(P, "F") * norm(inv(P), "F") * sqrt(sum(X^2))
D <- max(eigenM$values)
n <- log(0.01/C)/log(D)
C
```

```
## [1] 98782.31
```

```
D
```

```
## [1] 0.7498791
```

```
ceiling(n)
```

```
## [1] 56
```

#LARCH model

Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $x_0 = 0.1$ and $x'_0 = 1.1$.

```
C <- 1/sqrt(8*pi*exp(1))
D <- 0.5
n <- log(0.01/C)/log(D)
ceiling(n) + 1
```

```
## [1] 5
```

```
# get total variation estimate between iteration k of X and Y
# to be used in simulation of LARCH, asymmetric ARCH and GARCH models
getTv <- function(X,Y,k,binlength){
```

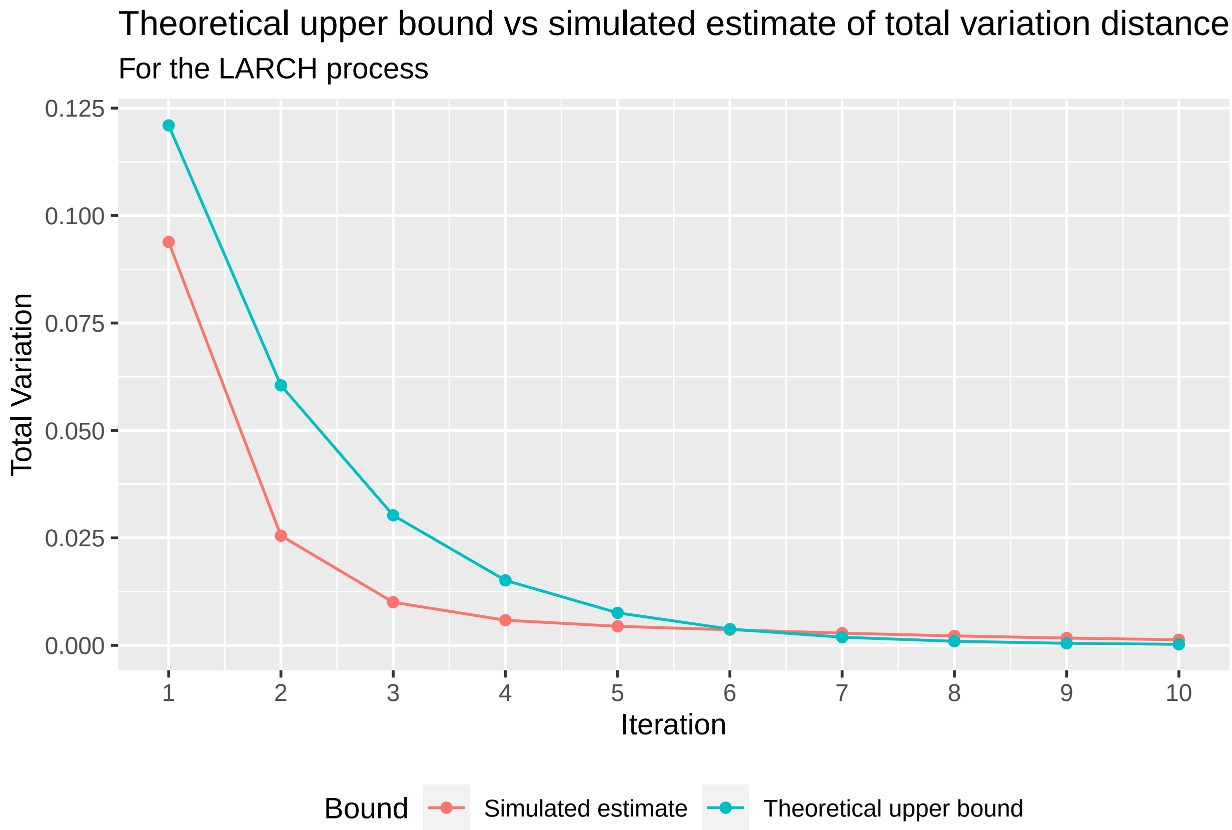
```r
  n <- dim(X)[1]
  maxVal <- max(X[,k],Y[,k])
  minVal <- min(X[,k],Y[,k])
  bins <- seq(from = minVal, to = maxVal+binlength, by = binlength)
  histX <- hist(X[,k], breaks=bins, plot = FALSE)
  histY <- hist(Y[,k], breaks=bins, plot = FALSE)
  diff <- abs(histX$counts-histY$counts)
  tv <- sum(diff)/(2*n)
  return(tv)
}
```

Code to generate figure 2.

```r
n <- 10^7 # no. of simulations
k <- 11 # no. of iterations
X <- matrix(0, nrow = n, ncol = k)
Y <- matrix(0, nrow = n, ncol = k)

# oX <- matrix(0, nrow = n, ncol = k)
# oY <- matrix(0, nrow = n, ncol = k)

a <- 1
b <- 0.5
X[,1] <- 0.1
Y[,1] <- 1.1


for (i in 1:n){
  Z <- rchisq(k, df=1)
  for(j in 2:k){
    X[i,j] <- (a + b*X[i,j-1]) * Z[j]
    Y[i,j] <- (a + b*Y[i,j-1]) * Z[j]
  }
}
```

```r
tv <- 1:k
for(i in 1:k){
  tv[i] <- getTv(X,Y,i, 0.01)
}
```

```r
df <- data.frame(Iteration = 1:(k-1), Actual = tv[2:k], Estimate = C * D^(0:(k-2)))

df %>%
  pivot_longer(Actual:Estimate, names_to = "Bound", values_to = "val") %>%
  ggplot(aes(x=Iteration, y=val, col=Bound)) +
  geom_line() +
  geom_point() +
  labs(fill = "Bound", y="Total Variation", title = "Theoretical upper bound vs simulated estimate of t
  theme(legend.position = "bottom") +
  scale_x_continuous(n.breaks = 10) +
  scale_colour_discrete(labels = c("Simulated estimate", "Theoretical upper bound"))
```



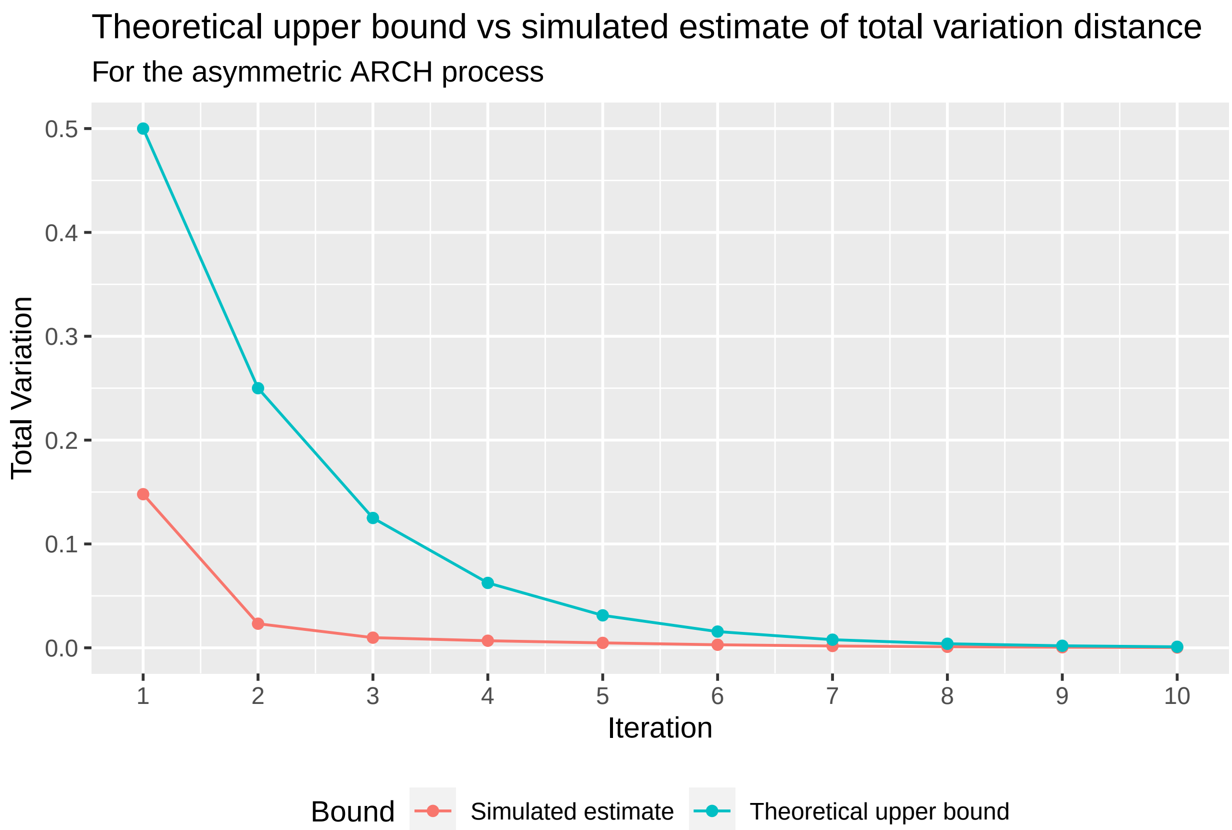
#Asymmetric ARCH model

Code to generate figure 3.

```
n <- 10^7 # no. of simulations
k <- 11 # no. of iterations
X <- matrix(0, nrow = n, ncol = k)
Y <- matrix(0, nrow = n, ncol = k)

a <- 0.5
b <- 3
c <- 4
X[,1] <- 0
Y[,1] <- 5

for (i in 1:n){
  Z <- rnorm(k)
  for(j in 2:k){
    X[i,j] <- sqrt((a*X[i,j-1]+b)^2 + c^2) * Z[j]
    Y[i,j] <- sqrt((a*Y[i,j-1]+b)^2 + c^2) * Z[j]
  }
}
```

```
tv <- 1:k
binlength <- 0.01
for(i in 1:k){
  tv[i] <- getTv(X,Y,i, binlength)
}
```

```
df <- data.frame(Iteration = 1:(k-1), Actual = tv[2:k], Estimate = 0.5^(1:(k-1)))

df %>%
  pivot_longer(Actual:Estimate, names_to = "Bound", values_to = "val") %>%
  ggplot(aes(x=Iteration, y=val, col=Bound)) +
  geom_line() +
  geom_point() +
  labs(fill = "Bound", y="Total Variation", title = "Theoretical upper bound vs simulated estimate of to
  theme(legend.position = "bottom") +
  scale_x_continuous(n.breaks = 10) +
  scale_colour_discrete(labels = c("Simulated estimate", "Theoretical upper bound"))
```



GARCH model using Intro to Timeseries example

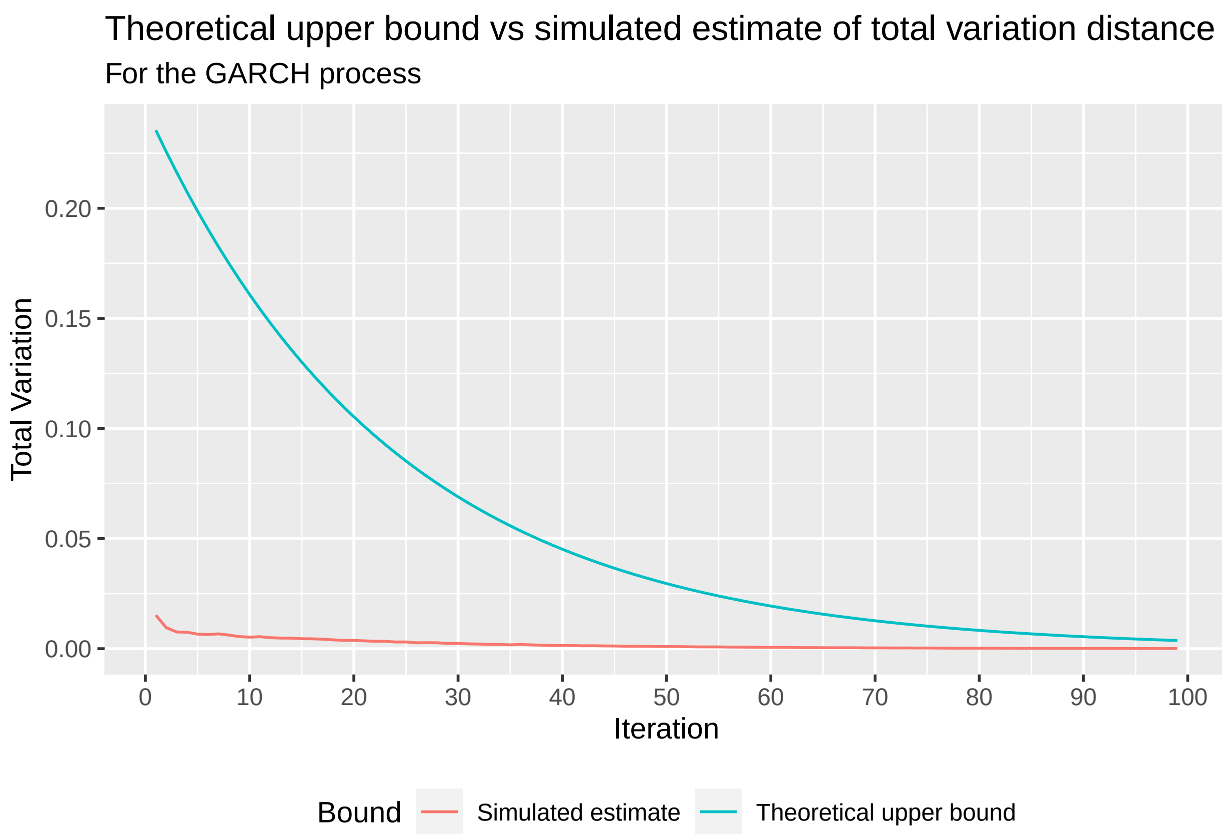
Code to generate figure 4.

```
n <- 10^6 # no. of simulations
k <- 100 # no. of iterations
X <- matrix(0, nrow = n, ncol = k)
Y <- matrix(0, nrow = n, ncol = k)

oX <- matrix(0, nrow = n, ncol = k)
oY <- matrix(0, nrow = n, ncol = k)
```

```
a <- 0.13
b <- 0.1266
c <- 0.7922
D <- sqrt(b+c)
X[,1] <- 0.1
Y[,1] <- -0.1
oX[,1] <- 0.01
oY[,1] <- 0.1
C <- sqrt( (b*abs(X[1,1]^2-Y[1,1]^2) + c*abs(oX[1,1]^2-oY[1,1]^2)) / (a * (b+c)) )
# oX[,1] <- 0.1
# oY[,1] <- 4

for (i in 1:n){
  Z <- rnorm(k)
  for(j in 2:k){
    oX[i,j] <- sqrt(a + b*X[i,j-1]^2 +c * oX[i,j-1]^2)
    oY[i,j] <- sqrt(a + b*Y[i,j-1]^2 +c * oY[i,j-1]^2)

    X[i,j] <- oX[i,j] * Z[j]
    Y[i,j] <- oY[i,j] * Z[j]
  }
}
```

```
tv <- 1:k
for(i in 1:k){
  tv[i] <- getTv(X,Y,i, 0.01)
}
```

```
df <- data.frame(Iteration = 1:(k-1), Actual = tv[2:k], Estimate = C * D^(2:k))

df %>%
  pivot_longer(Actual:Estimate, names_to = "Bound", values_to = "val") %>%
  ggplot(aes(x=Iteration, y=val, col=Bound)) +
  geom_line() +
  labs(fill = "Bound", y="Total Variation", title = "Theoretical upper bound vs simulated estimate of t
  theme(legend.position = "bottom") +
  scale_x_continuous(n.breaks = 10) +
  scale_colour_discrete(labels = c("Simulated estimate", "Theoretical upper bound"))
```



Calculation to find the number of iterations needed to guarantee a TV of 0.01 when $x_0 = 0.1, \sigma_0 = 0.01$ and $x'_0 = -0.1, \sigma'_0 = 0.1$.

```
n <- log(0.01/C)/log(D)
ceiling(n)
```

```
## [1] 77
```