Report of Lab 6: Simulation of Stochastic Processes

Alberto Nieto Sandino

October 25, 2018

1 Introduction

In this lab, we are gonna focus on how to simulate the sample path of stochastic processes.

Firstly, we will simulate a shot noise process and see the different approaches that can be taken, and why certain methods are more preferable than others when it comes to estimating parameters and their confidence intervals.

Secondly, we will simulate a Gaussian process and see the behaviour it has for a certain time sequence.

2 Assignment 1

2.1 Problem

Our first task is to simulate the sample path of a shot noise process $\{X(t)\}_{t\in[0,10]}$ with $\lambda = 1$. It can be expressed as

$$g(t) = \begin{cases} 0 & \text{for } t < 0 \\ 1 & \text{for } 0 \leq t \leq \frac{1}{2} \\ 0 & \text{for } \frac{1}{2} < t. \end{cases} \tag{1}$$

As a help, this process $X(t)$ can be thought of as modelling the number of vehicles on a bridge at a time t with inter arrival times for vehicles as $\exp(\lambda)$ distributed. The vehicles need $\frac{1}{2}$ units of time to cross the bridge.

2.2 Theory and implementation

We are to simulate the sample path of a process where the signal package $g(t)$ arrives with inter arrival times that are $\exp(\lambda)$ distributed. The total signal produced by this type of process is called *shot noise* and can be defined as

$$X(t) = \sum_{k=1}^{\infty} g\left(t - \sum_{j=1}^{k} \xi_j\right) + \sum_{k=1}^{\infty} g\left(t + \sum_{j=1}^{k} \eta_j\right) \quad \text{for } t \in \mathbb{R}. \tag{2}$$

Based on definition 6.7, $\xi_1, \xi_2, ..., \eta_1, \eta_2, ...$ are independent $\exp(\lambda)$ distributed random variables and g is a function $g : \mathbb{R} \to \mathbb{R}$ that satisfies $\int_{-\infty}^{\infty} |g(x)|dx < \infty$.

Another way to express shot noise is by the relation

$$X(t) = \int_{-\infty}^{\infty} g(t-s)dY(s) + \int_{-\infty}^{\infty} g(t+s)dZ(s), \tag{3}$$

where $\{Y(s)\}_{s\geq 0}$ and $\{Z(s)\}_{s\geq 0}$ are independent Poisson processes with intensities λ. By the aforementioned facts, it can be proven that shot noise is *weakly stationary* with expectation function $m_X = \lambda \int_{-\infty}^{\infty} g(s)ds$ and covariance function $r_X(\tau) = \lambda \int_{-\infty}^{\infty} g(s)(\tau+s)dx$.

The expectation function $m_X(t)$ gives us information about the one-dimensional marginal distribution $F_{X(t)}(x) = \mathbf{P}\{X(t) \leq x\}$, $t \in T$ of the process $X(t)$. While, the covariance function $r_X(s,t)$ gives us information about the two-dimensional distributions $F_{X(s),X(t)}(x,y) = \mathbf{P}\{X(s) \leq x, X(t) \leq y\}$, $s,t \in T$ of the process $X(t)$.

By definition 6.6, we know that since the expectation function $m_X(t)$ and the covariance function $r_X(t, t+\tau)$ do not depend on $t \in T$ (which is the case for shot noise), the process is weakly stationary. Thus, we can refer to $m_X(t)$ as m_X, and $r_X(t, t+\tau)$ as $r_X(\tau)$.

Given the form of the function $g(t)$, we consider $\xi_1, ..., \xi_{K_1}$ with K_1 being the largest integer such that the cumulative sum of ξ_i is smaller or equal to 10, in order to simulate the sample path $\{X(t)\}_{t\in[0,10]}$. This is because ξ_i represents the total number of cars that are simultaneously on the bridge at time t. Since $g(t)$ is only non-zero in $t \in [0, 1/2]$, it is not possible for $g(t)$ to be 1 if the cumulative sum of ξ_i is strictly greater than 10.

Moreover, we consider $\eta_1, ..., \eta_{K_2}$ with K_2 being the largest integer such that the cumulative sum of η_i is smaller or equal than 0.5. This is because η_i represents the total number of cars that are on the bridge at a time $t = 0$. Since $g(t)$ is only non-zero in $t \in [0, 1/2]$, it is not possible for $g(t)$ to be 1 if the cumulative sum of η_i is strictly larger than $1/2$ at time $t = 0$.

After these considerations we can rewrite equation (2) as

$$X(t) = \sum_{k=1}^{K_1} g\left(t - \sum_{j=1}^{k} \xi_j\right) + \sum_{k=1}^{K_2} g\left(t + \sum_{j=1}^{k} \eta_j\right) \quad \text{for } t \in [0, 10]. \tag{4}$$

The limits for the calculations of the sample path of $X(t)$ are reduced to K_1 and K_2, thus resulting in much faster computation time since all the other summation components were zero.

When it came to the implementation in R, we started by defining values that $g(t)$ takes for time t. Then, we started to build the shot noise process by defining the exponentially distributed (`rexp(1,rate=lambda)` random variables ξ_i and η_i. This was done by generating random deviates from an exponential distribution until the cumulative sum (`cumsum`) of the variable (both for ξ_i and η_i) reached their upper limit.

After this, we built the shot noise process function as a function of time t in the way defined by equation (4). We evaluated the shot noise process in the time interval of interest $t \in [0, 10]$ and `plot` it using `stepfun` to interpolate the function points.

2.3 Results and discussion

The results of simulating the sample path of a shot noise process $\{X(t)\}_{t\in[0,10]}$ with $\lambda = 1$ can be seen in Figure 1. We can see that the maximum number of cars that are simultaneously passing through the bridge is 4, which occurs twice in this time $t \in [0, 10]$. Furthermore, we can observe that the time it takes to cross the bridge is, indeed, 1/2 time units.

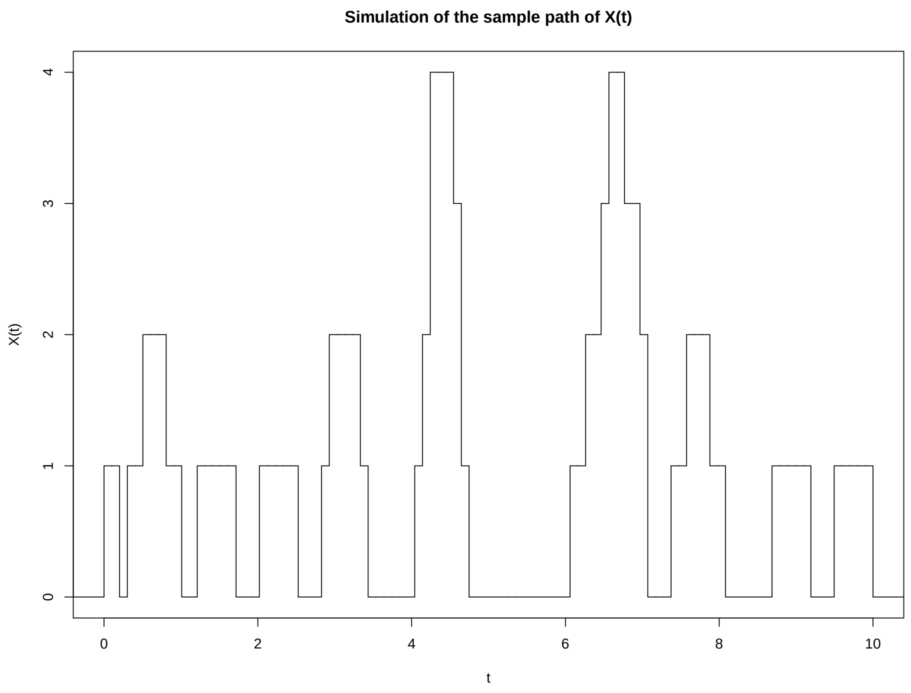

Figure 1: Simulation of the sample path for a shot noise process

Assignment 2

3 Assignment 2.1

3.1 Problem

Our task is to estimate the confidence interval of $p = \mathbf{P}\{\max_{t\in[0,10]} X(t) \geq 3\} = \mathbf{E}\{\zeta\}$ where $\zeta = 1$ if $\max_{t\in[0,10]} \geq 3$. ζ is a random variable and the confidence interval is estimated over 10000 simulations.

3.2 Theory and implementation

In order to simulate ζ, it is convenient to follow the procedure described in the *hint* given in the assignment. We can write

$$\mathbb{M} \equiv \{-(\eta_1 + ... + \eta_{K_2}), ..., -(\eta_1 + \eta_2), -\eta_1, \xi_1, \xi_1 + \xi_2, ..., \xi_1 + ... + \xi_{K_1}\} \tag{5}$$

with K_1 (and K_2), the largest integers such that the cumulative sum of ξ_i (and η_i) is smaller or equal to 10 (and 0.5). Then, we can obtain

$$X(t) = \sum_{k=1}^{\infty} g\left(t - \sum_{l=1}^{k} \xi_l\right) + \sum_{k=1}^{\infty} g\left(t + \sum_{l=1}^{k} \eta_l\right) \quad \text{for } t \in [0, 10], \tag{6}$$

if and only if there exist some $m_1, m_2, m_3 \in \mathbb{M}$ which accomplish that

$$-\frac{1}{2} \leq m_1 < m_2 < m_3 \leq 10 \text{ for } m_3 - m_1 \leq \frac{1}{2}. \tag{7}$$

To sum up, we simulate the $\exp(\lambda)$ distributed random variables $\xi_1, ..., \xi_{K_1}, \eta_1, ..., \eta_{K_2}$, and check that the requirements specified in the expression (7) are met given $\mathbb{M}$ from equation (5). If in the i-th simulation, the conditions are met; then, $\max_{t\in[0,10]} X(t) \geq 3$, so ζ is set to 1. Otherwise, ζ is set to zero.

A given set of m_1, m_2, m_3 that meet the requirements in the expression (7), produce $(t - m_i)$ and $(t + m_i)$ in the interval $[0, 1/2]$. Then, we know that $g(t) = 1$ for $t \in [0, 1/2]$ so $g(t)$ will take the value of 1 for the given set of m_i. Thus, for a time $t \in [0, 10]$, when $g(t) = 1$ for some m_i, it gives $X(t) \geq 3$ as can be seen in equation (6).

The whole procedure in R can be summarized as a `for` loop where the 10000 simulations are performed in order to find a value of ζ to estimate $\hat{p}$. Inside the loop, ξ_i and η_i are constructed as it was done in the previous assignment. Then, the values of these random variables that meet the cumulative sum requirement (i.e. `cumsum` $(\eta) \leq 0.5$ and `cumsum` $(\xi) \leq 10$) are used to construct $\mathbb{M}$. We check that the expression (7) is valid for our values in $\mathbb{M}$, and if it is true, ζ is set to 1. Otherwise, it is set to zero.

To estimate p, we know that $p = \mathbf{E}\{\zeta\}$. So we solve for $\hat{p}$ by finding the `mean` of the values of ζ from the simulations.

Once we know $\hat{p}$, we can find the normal 95% confidence interval by calculating $[\hat{p} - z_{1-\alpha/2}\hat{se}, \hat{p} - z_{\alpha/2}\hat{se}]$, where $\alpha = 0.05$ and $\hat{se}$ is the standard error calculated as $\hat{se} = \frac{\hat{\sigma}}{\sqrt{n}}$. The quantiles $z_{1-\alpha/2}$ and $z_{\alpha/2}$ are calculated using `qnorm`.

3.3 Results and discussion

After 10000 simulations, we obtain an estimation of $p = 0.4936$ with normal confidence interval CI = $[0.4838, 0.5034]$. The confidence interval is small, which speaks of the accuracy of the simulation. The accuracy could be improved by increasing the number of simulations.

4 Assignment 2.2

4.1 Problem

Our task now is the same as for Assignment 2.1. However, we will take a different approach. In this case, we will use the same strategy as in Assignment 1 and count the number of times that our plot reaches values higher than 2. We will also examine why this obvious solution might not be as good as it seems.

4.2 Theory and implementation

As in assignment 1, we simulate the sample path of a shot noise process $X(t)$. We start with the definition of the function $g(t)$. Then, we defined the $\exp(\lambda)$ distributed random variables (ξ_i and η) by `rexp`. We checked that their cumulative sums (`cumsum`) did not cross the limit set for each of them.

After that, we built the shot noise process function as defined in equation (4) and we evaluated the function in the time $t \in [0, 10]$. For these result, we checked if the plot has greater than 2 and if it was the counter was set to 1.

This algorithm was simulated 10000 times and the estimated $\hat{p}$ was found as the `mean` of the number of hits. The normal confidence interval of the estimate is calculated exactly in the same way as it was described in Assignment 2.1.

4.3 Results and discussion

Using this simple method to estimate $\hat{p}$ and the confidence intervals gives the results shown in Table 1 for the different values of the time increment h. Compared to the results we obtain from the hint, this method needs very small time increments to achieve similar results (i.e. $h \approx 0.01$). However, when we look at the CPU time, the time needed for computing the simulations with such a h is much bigger than for the method in Assignment 2.1. Indeed, we can see that as h decreases, the CPU time increases exponentially.

Table 1: Results of the estimate and confidence interval of $\hat{p}$ for different time increments h

Time increment (h)	Estimate ($\hat{p}$)	CI	CPU time [s])
1	0.1521	$[0.1451, 0.1591]$	1.99
0.1	0.446	$[0.4363, 0.4557]$	15.10
0.01	0.4947	$[0.4849, 0.5045]$	145.83
0.001	0.4826	$[0.4728, 0.4924]$	1378.43

5 Assignment 3

5.1 Problem

Our final task is to simulate the sample path for $\{X(t)_{t\in[0,10]}\}$ where $X(t)$ is a *stationary Gaussian process* of the form

$$X(t) = \int_{-\infty}^{\infty} f(t+s)dW(s) \quad \text{for} \quad t \in \mathbb{R} \tag{8}$$

where $\{W(s)\}_{s\in\mathbb{R}}$ is a Wiener process and $f(t)$ has the form

$$f(t) = \begin{cases} 1 - t^2 & \text{for} \quad |t| \leq 1 \\ 0 & \text{for} \quad |t| > 1 \end{cases} \tag{9}$$

5.2 Theory and implementation

A Gaussian process, or normal process, $\{X(t)_{t\in\tau}\}$ are characterized by the linear combination of the process values $\sum_{k=1}^{n} a_k X(t_k)$ which are normally distributed for any $n \in mathbbN$ and any $a_1, ..., a_n, t_1, ..., t_n \in \mathbb{R}$. This means that linear filters with Gaussian input produce Gaussian output. Gaussian processes can be characterized by its mean and covariance function.

A Wiener process is a Lvy process with $W(0) = 0$ and $W(t+h) - W(t)$ that follows a normal distribution $N(0, h)$ for $t \in \mathbb{R}$ were the increments of $W(t)$ are independent (if and only if they are uncorrelated).

A stationary Gaussian process can be approximated by

$$X(t) \approx \sum_{k=-s(n)}^{s(n)} f\left(t + \frac{k}{n}\right)\left(W\left(\frac{k+1}{n}\right) - W\left(\frac{k}{n}\right)\right), \tag{10}$$

where the summation limits $\pm s(n)$ satisfy that $\lim_{n\to\infty} s(n)/n = \infty$.

The function $f(t)$ was defined first according to equation (9). Then, we defined the number of increments `n`, the time vector `x` and the time increment `h` as $h = t/n$. We also defined `s(n)` as the squared value of the number of increments `n` as to comply with the aforementioned limit tending to infinity as n increases.

The Gaussian process was defined as a function describing equation (10) with the increment `inc` defined as a random variable normally distributed with zero-mean. Then, it was evaluated for the time $t \in \mathbb{R}$ and plotted it by `plot`.

5.3 Results and discussion

The simulation for the sample path of the Gausssian process $X(t)_{t\in[0,10]}$ can be found in Figure 2.

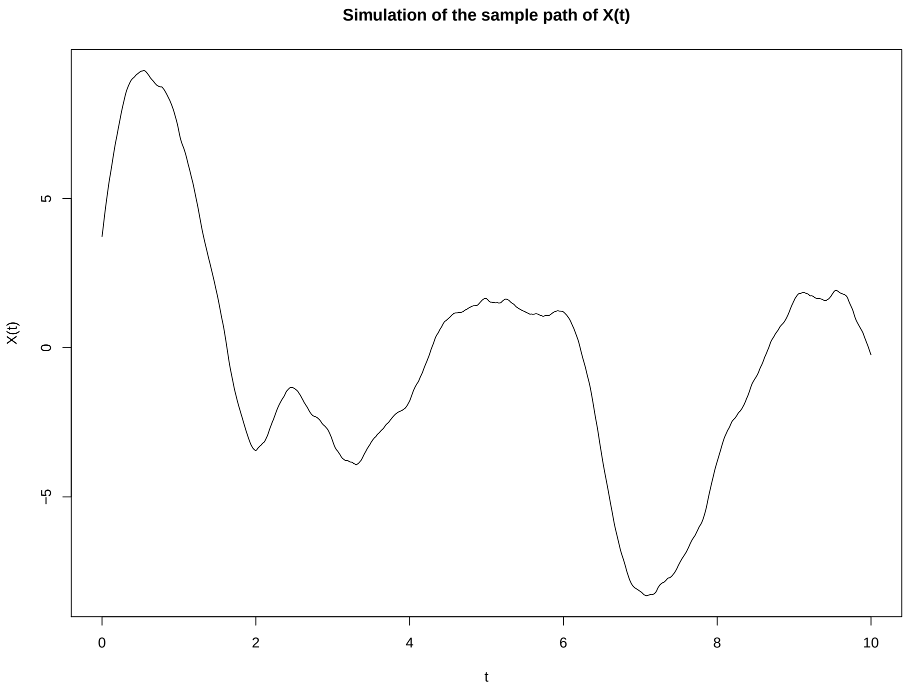

Figure 2: Simulation of the sample path of a Gaussian process for $t \in \mathbb{R}$

Appendix - R code

```r
# Lab 6 - Simulation of stochastic processes
# Author: Alberto Nieto Sandino
# Date: 2018-10-23

# Clean environment
closeAllConnections()
rm(list=ls())
# Clear plots
graphics.off()
# Clear history
clearhistory <- function() {
  write("", file=".blank")
  loadhistory(".blank")
  unlink(".blank")
}
clearhistory()

# Example 6.3 - Simulation of a Poisson process
x <- cumsum(rexp(50))
y <- cumsum(c(0,rep(1,50)))
plot(stepfun(x,y),xlim = c(0,10), do.points = F)

# Example 6.4 - Simulation of a sample path of a Poisson process
n <- 100
x <- seq(0,10,length = 1000)
y <- cumsum(rpois(1000,1/n))
plot(x,y)

# Example 6.2 - Simulation of a sample path for a standard Wiener
   process
h <- 1/1000 # step size
t <- seq(0,1,by = h)
inc <- rnorm(1000, mean = 0, sd = sqrt(h))
W <- c(0,cumsum(inc))
plot(t,W,type='l')

##################
## Assignment 1 ##
##################

# Clean environment
closeAllConnections()
rm(list=ls())
```

```r
# Clear plots
graphics.off()

lmd = 1 # rate of exponential distribution
t <- seq(0,10, length = 100)

# Define g(t)
g <- function(a){
  if ((a >= 0) && (a <= (1/2))){ # values of g(t) go from 0 to 0.5
    return(1)
  } else{
    return(0)
  }
}

# Because of the form of function g(t), we need to define xi and eta

# form xi so that cdf(xi) is equal or smaller than 10
xi <- rep(0,20)
i <- 1
while (cumsum(xi)[i] < 10){
  xi[i] = rexp(1, rate = lmd)
  i = i + 1
}
xi[i] <- 0

# form eta so that cdf(eta) is equal or smaller than 0.5
eta <- rep(0,10)
j <- 1
while (cumsum(eta)[j] < 0.5){
  eta[j] = rexp(1,rate = lmd)
  j = j + 1
}
eta[j] <- 0

# Construct the function to calculate the sample path
shot_noise <- function(t){
  X <- 0 # initialize X
  for (k in 1:(i-1)){
    X = X + g(t-cumsum(xi)[k]) # sumation of xi components
  }
  for (n in 1:(j-1)){
    X = X + g(t+cumsum(eta)[n]) # sumation of eta components
  }
  return (X) # return the computation of shot noise
```

```r
}

y <- rep(0,length(t))
for (l in 1:length(t)){
  y[l] <- shot_noise(t[l]) # Sample path for the specified time, t
}

plot(stepfun(t, c(0,y)), xlim = c(0,10), do.points = F,
     xlab = "t", ylab = "X(t)", main = "Simulation of the sample
        path of X(t)")


##################
## Assignment 2 ##
##################

# Clean environment
closeAllConnections()
rm(list=ls())
# Clear plots
graphics.off()


####################
## Assignment 2.1 ##
####################

lmd <- 1
n <- 10000

zeta <- rep(0,n)
for (i in 1:n){ # Perform 10000 simulations

  # Constructing xi
  xi <- rep(0,100)
  j <- 1
  while (cumsum(xi)[j]<10){
    xi[j] <- rexp(1, rate = lmd)
    j <- j + 1
  }
  xi[j] <- 0

  # Constructing eta
  eta <- rep(0,100)
  k <- 1
  while (cumsum(eta)[k]<0.5){
    eta[k] <- rexp(1,rate = lmd)
    k <- k + 1
  }
  eta[j] <- 0

  # Constructing M (eq. 6.3)
  M_1 <- -cumsum (eta)[cumsum(eta) < 0.5]
  M_2 <-  cumsum (xi) [cumsum(xi)  < 10]
  M   <- sort(c(M_1, M_2))

  # Check that condition in eq (6.4) holds for M (TRUE-> zeta = 1)
  if (length(M) > 2){
    for (l in 3:length(M)){
      if (M[l] - M[l-2] <= 0.5){
        zeta[i] <- 1
      }
    }
  }
}

# Expected value of zeta (mean)
p_hat <- mean (zeta)
p_hat
# Standard error of zeta
sigma_p <- sd(zeta)
se_hat <- sigma_p/sqrt(n)
se_hat
# Calculate the normal CI
alpha <- 0.05
z.lower <- qnorm (alpha/2) # lower limit
z.upper <- qnorm (1-(alpha/2)) # upper limit
ci <- c(p_hat + z.lower*se_hat,
        p_hat + z.upper*se_hat)
ci

####################
## Assignment 2.2 ##
####################

# Clean environment
closeAllConnections()
rm(list=ls())
# Clear plots
graphics.off()
```

```

hits <- rep(0,10000) # initialize counter for hits
lmd = 1 # rate of exponential distribution
h <- 1/100 # distance between time points
t <- seq(0,10, by = h)

# Define g(t)
g <- function(a){
  if ((a >= 0) && (a <= (1/2))){ # values of g(t) go from 0 to 0.5
    return(1)
  } else{
    return(0)
  }
}

start_time <- Sys.time() # Start measuring time
for (n in 1:10000){

  # Because of the form of function g(t), we need to define xi and
     eta
  # form xi so that cdf(xi) is equal or smaller than 10
  xi <- rep(0,200)
  i <- 1
  while (cumsum(xi)[i] < 10){
    xi[i] = rexp(1, rate = lmd)
    i = i + 1
  }
  xi[i] <- 0

  # form eta so that cdf(eta) is equal or smaller than 0.5
  eta <- rep(0,100)
  j <- 1
  while (cumsum(eta)[j] < 0.5){
    eta[j] = rexp(1,rate = lmd)
    j = j + 1
  }
  eta[j] <- 0

  # Construct the function to calculate the sample path
  shot_noise <- function(t){
    X <- 0 # initialize X
    for (k in 1:(i-1)){
      X = X + g(t-cumsum(xi)[k]) # sumation of xi components
    }
    for (n in 1:(j-1)){
      X = X + g(t+cumsum(eta)[n]) # sumation of eta components
    }
    return (X) # return the computation of shot noise
  }

  y <- rep(0,length(t))
  for (l in 1:length(t)){
    y[l] <- shot_noise(t[l]) # Sample path for the specified time, t
    if (y[l] > 2){
      hits[n] <- 1
    }
  }
}

# Expected value of zeta (mean)
p_hat <- mean (hits)
p_hat
# Standard error of zeta
sigma_p <- sd(hits)
se_hat <- sigma_p/sqrt(n)
se_hat
# Calculate the normal CI
alpha <- 0.05
z.lower <- qnorm (alpha/2) # lower limit
z.upper <- qnorm (1-(alpha/2)) # upper limit
ci <- c(p_hat + z.lower*se_hat,
        p_hat + z.upper*se_hat)
ci

end_time <- Sys.time() # End of measured time
cpu_time <- end_time - start_time
cpu_time


##################
## Assignment 3 ##
##################

# Clean environment
closeAllConnections()
rm(list=ls())
# Clear plots
graphics.off()

# We start by defining f(t)
```

```r
f <- function(t){
  fun <- 1 - t^2
  fun [abs(t) > 1] <- 0
  return (fun)
}

n <- 1000 # number of increments
t <- 10
h <- t/n # distance between time points
x <- seq(0,t, by = h)
s_n <- n^2 # Define the summation limits

# Define function to apporximate a stationary Gaussian process X(t)
gaussian_process <- function (t){
  res <- 0
  for (k in (-s_n):s_n){
    inc <- rnorm (1, mean = 0, sd = sqrt(h))
    res = res + f(t+k/n)*inc
  }
  return (res)
}

y = gaussian_process(x)
plot(x, y, xlim = c(0,10), type = 'l',
     xlab = "t", ylab = "X(t)", main = "Simulation of the sample
        path of X(t)")
```