Oblique Reflected Diffusion Simulation with Penalty Method

Charles Amponsah, Andrey Sarantsev

28 March, 2019

Introduction

Given a stochastic differential equation (SDE) with oblique reflection :

$$dX(t) = g(X(t))dt + \sigma(X(t))dW(t) + r(X(t))d\ell(t) \tag{1}$$

with $\ell(t)$ is a continuous nondecreasing process with $\ell(0) = 0$ which can increase only when it inside it domain and reflected back inside its domain in the direction of r(z) as X hits the domain at a point z. Once the process is in it domain, $X(t)$ behaves as a solution to the SDE below.

$$dX(t) = g(X(t))dt + \sigma(X(t))dW(t). \tag{2}$$

We implement the numerical approximation of solutions of stochastic differential equations driven by brownian motion which are moving inside a domain and obliquely reflected at the boundary as follows:

$$z_{k+1} - z_k = [g(z_k) + \varphi(d(z_k)) \times r(y(zk))]\epsilon + \sigma(z_k)\epsilon_{k+1}; \tag{3}$$

where $\epsilon_1, \epsilon_2, ... \sim N(0; \epsilon I_d)$. We define this function Z_ϵ as piecewise constant: $Z_\epsilon(t) := Z_\epsilon(k\epsilon) = z_k, t \in [k\epsilon; (k+1)\epsilon), k = 0, 1, ...$

Half-line

Obliquely reflected brownian motion with penalty term

We simulate a brownian motion with a penalty term for half-line.

$$z(t+\epsilon) \approx z(t) + [\varphi(d(z(t))) \times r(y(z(t)))]\epsilon + \varepsilon_{t+\epsilon},$$

where $Z(0) = 1, \quad \varepsilon_{t+\epsilon} \sim \mathcal{N}(0, \epsilon), \epsilon = 0.01$, time horizon $t=10, a=100, $ and

$$\varphi(d(z(t))) \times r(y(z(t))) = \textit{ penalty term with reflection} = \begin{cases} 0, & if \quad z(t) \geq 0 \\ a[z(t)]^p, if & z(t) < 0 \end{cases}$$

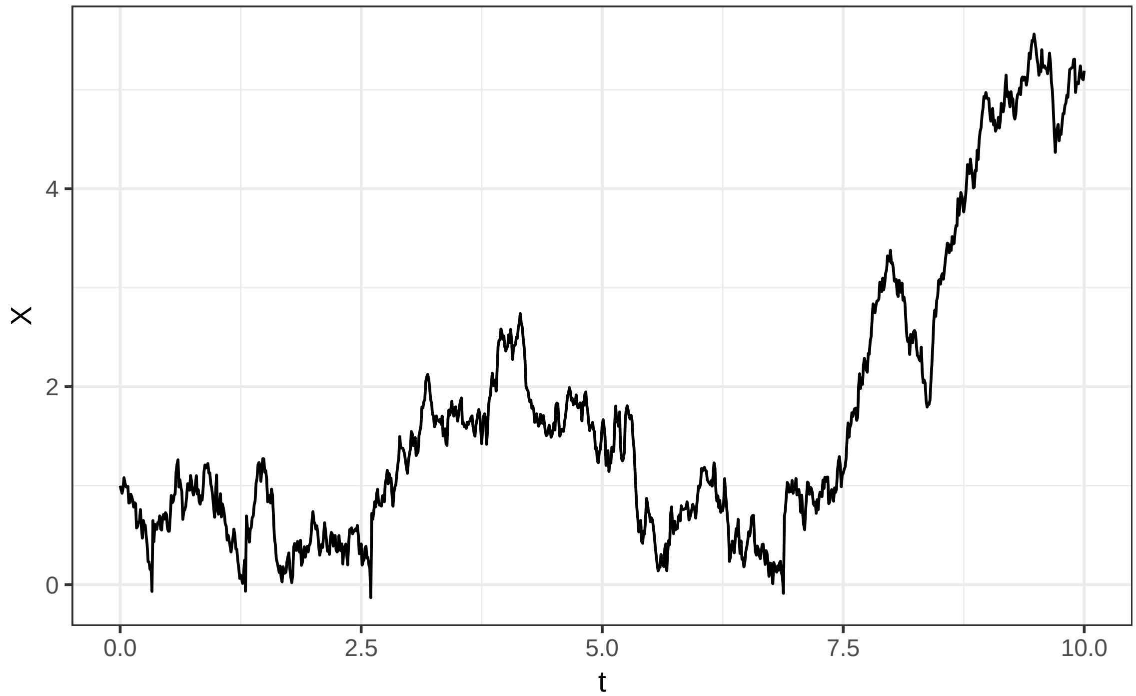


Figure 1: Obliquely reflected brownian motion with penalty

Convergence on half-line

A reflected brownian motion for unit negative drift has a stationary distribution, which is exponential with mean $\frac{1}{2}$, which serves as a limiting distribution for a long-time limit. See figure (1) below for $a = 100$

2-D Brownian motion with $\epsilon = 0.01, X(0) = 1$ *and time horizon* $t = 10$

$$Z(\vec{t + \epsilon}) - Z(\vec{t}) \sim \mathcal{N}(\vec{0}, \Sigma)$$

where $\Sigma = \begin{pmatrix} \epsilon & 0 \\ 0 & \epsilon \end{pmatrix}$ and $Z(\vec{.})$ is a BM with 2-D

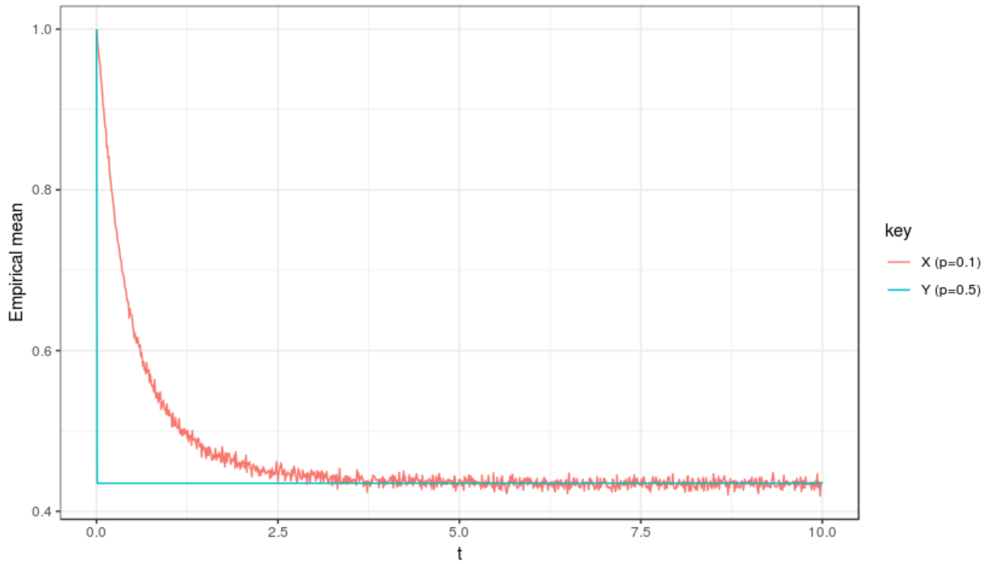
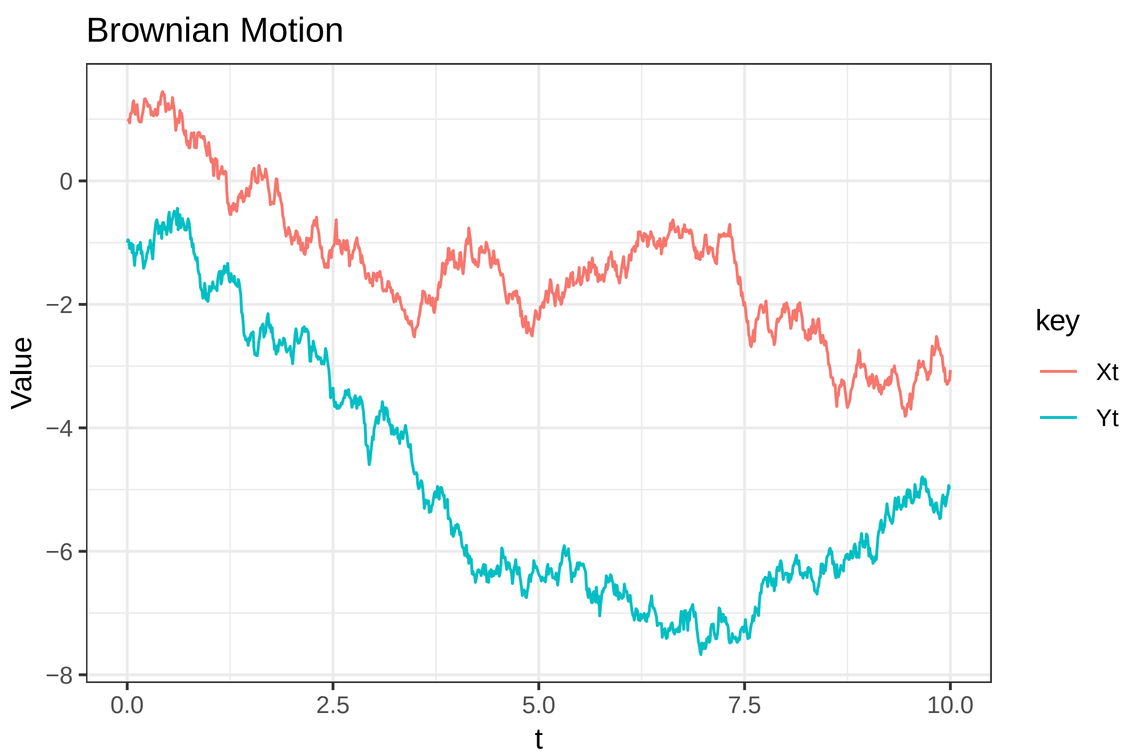
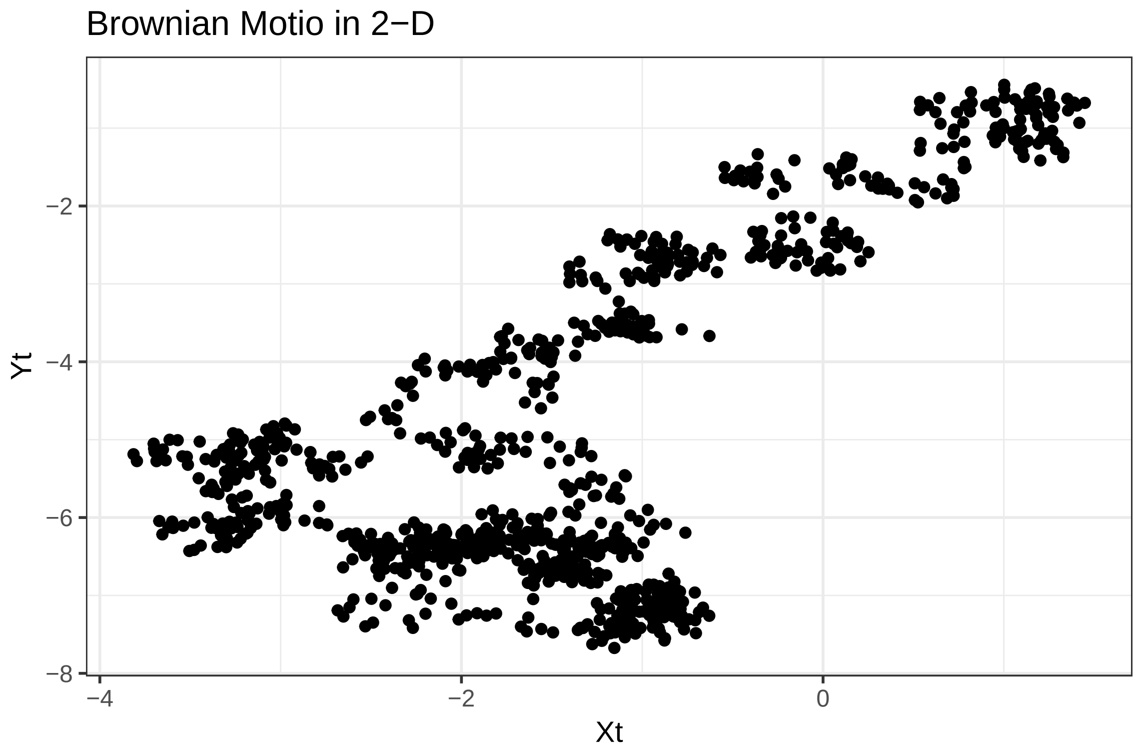


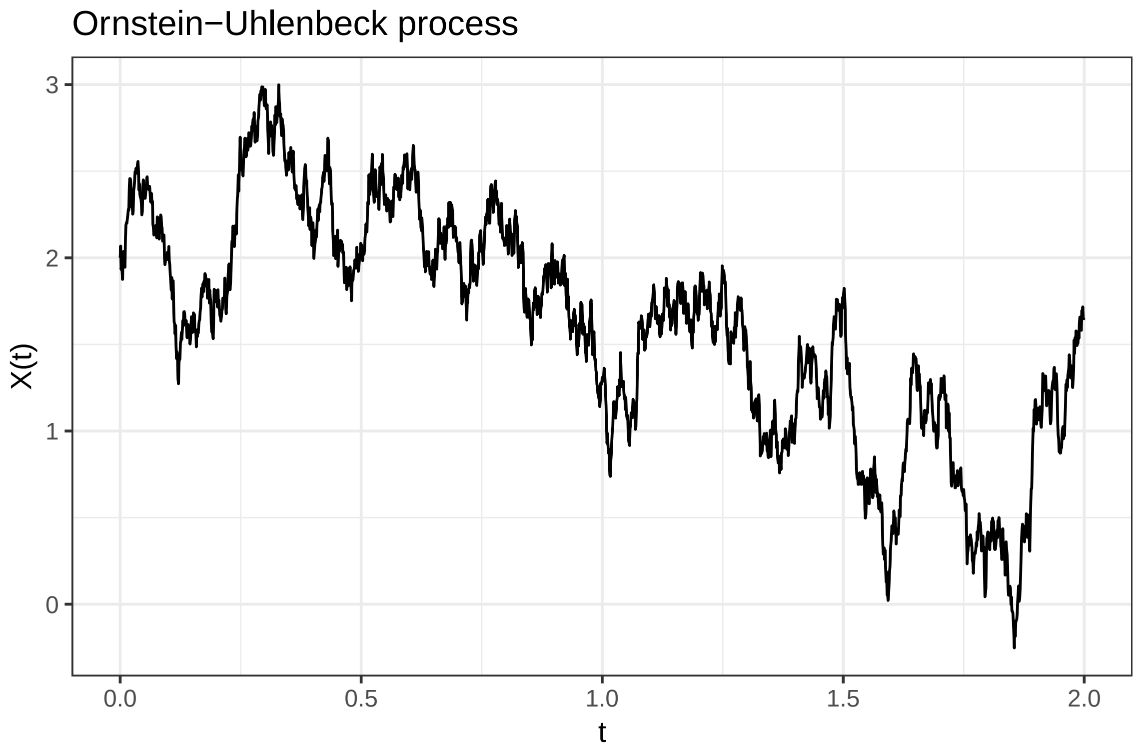
Figure 2: Stationary distribution of reflected brownian motion with unit negative drift



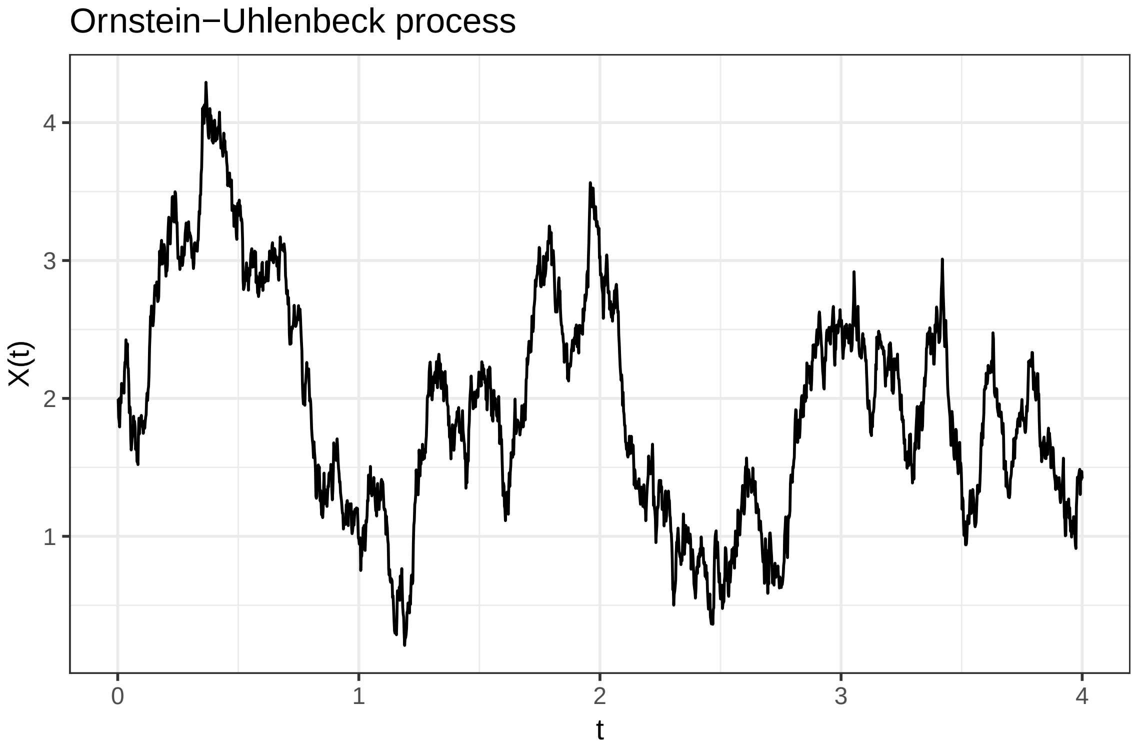


Ornstein-Uhlenbeck process with $\epsilon = 0.001, X(0) = 2$ ***and time horizon*** $t = 2$

$$dX(t) = 3(2 - X(t))dt + 2dW(t)$$

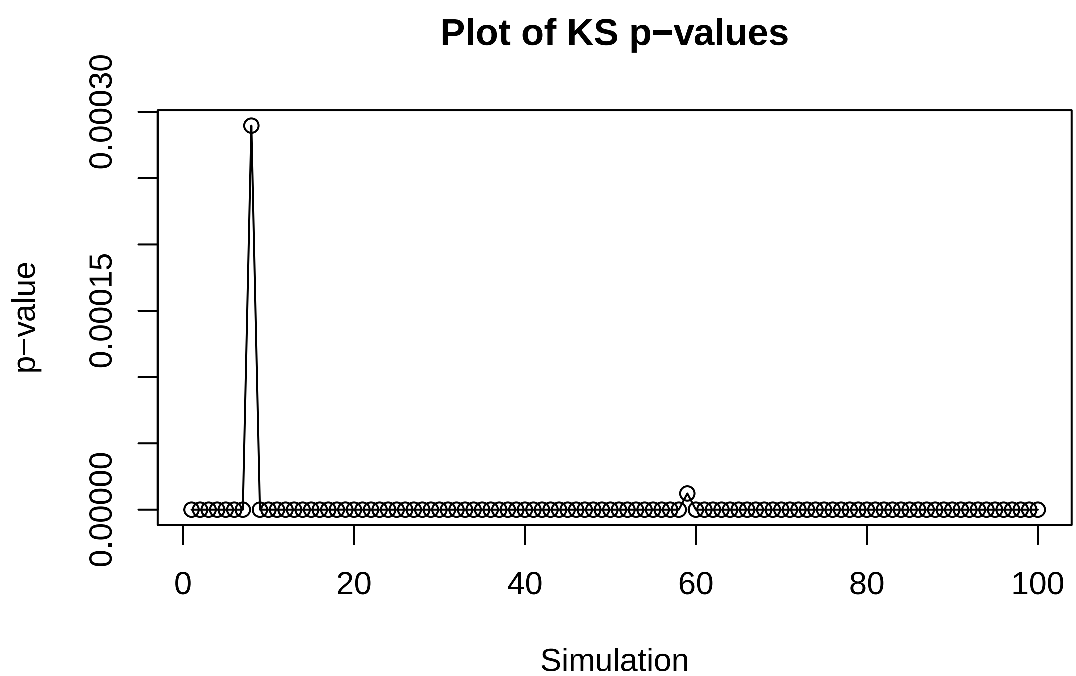

Ornstein−Uhlenbeck process
3
2
1
0
X(t)
0.0
0.5
1.0
1.5
2.0
t

Ornstein-Uhlenbeck process repeated with step $\epsilon = 0.002, X(0) = 2$ ***and time horizon*** $t = 2$




```
[1] 2.533675
```

Comparing Ornstein-Uhlenbeck processes using KS test



Two-dimensional SDE

Comparing 2-D SDE processes using Peacok test

R Code

```
rm(list = ls())
knitr::opts_chunk$set(
  echo = FALSE,      # don't show code
  comment = NA,
  error=FALSE ,
  warning = FALSE,   # don't show warnings
  message = FALSE,   # don't show messages (less serious warnings)
  cache = FALSE,     # set to TRUE to save results from last compilation
  fig.align = "center",# center figures
  fig.pos='h'
```

```r
)

## libraries
library(MASS)
library(dplyr)
library(tidyr)
library(ggplot2)
library(Peacock.test)


### BM with reflection: drift=0, sigma=1, X0=1 and penalty
SD_RF <- function(begin, end,step=0.01,a=100,p=0.1,r=1, X0=0){
  t<-seq(begin,end,step) ### create sequence
  n=length(t)
  dt  <- step
  dw<-rnorm(n, 0, sqrt(step))
  X<-NULL
  X[1] <- X0
  ## penalty function
penalty<- function(a,p,r,x){
 ifelse(x>=0,0,a*((abs(x))^p))
}
  for (i in 2:n) {
     X[i] <-  X[i-1] + penalty(a,p,r,X[i-1])*dt + dw[i]
  }
  return(data.frame(t,X));
}

X1<-SD_RF(0,10,step = 0.01,X0=1)
X1%>%gather(key,value, X) %>%
    ggplot(aes(x=t, y=value)) +
    geom_line() +theme_bw()+
  ggtitle("") +
  xlab("t") + ylab("X")

knitr::include_graphics(
  "C:/Users/Charles/Documents/UNR/BOOKS/SEM8/Stochastic Simulation/Rplot-Exponential")
### function to simulate multidimentional Brownian Motion
Multi_BrownianM<-function(Begin, End,Step, X0){
  t<-seq(Begin,End,Step)
  n=length(t)-1
  mu <- c(0,0) # Mean
  sigma <- matrix(c(Step, 0, 0, Step),2)
  Xt<-apply(rbind(X0, mvrnorm(n, mu = mu, Sigma = sigma )),2,cumsum)
```

```
  colnames(Xt) <- c("Xt","Yt")
  return(data.frame(t,Xt))
}

X2<-Multi_BrownianM(0,10,Step = 0.01,X0=c(1,-1))

par(mfrow=c(1,2))
X2%>% gather(key,value, Xt, Yt) %>%
    ggplot(aes(x=t, y=value, colour=key)) +
    geom_line() +theme_bw()+
  ggtitle("Brownian Motion") +
  xlab("t") + ylab("Value")

ggplot(data = X2, aes(x=Xt, y=Yt)) +
    geom_point() +theme_bw()+
  ggtitle("Brownian Motio in 2-D") +
  xlab("Xt") + ylab("Yt")

#plot(X1$Xt,X1$Yt, lwd=2,xlab="X(t)", ylab="Y(t)",
#     main="Brownian Motio in 2-D")

### simulate ORNSTEIN-UHLENBECK  Process
#Mu=long run mean, lambda = mean reversion speed sigma=
#set.seed(12345)
ornstein_uhlenbeck <- function(Begin, End,Step,Mu,lambda,sigma,X0){
  t<-seq(Begin,End,Step) ### create sequence
  n=length(t)
  dt  <- Step
  dw<-rnorm(n, 0, sqrt(Step))
  X<-NULL
  X[1] <- X0
  for (i in 2:n) {
    X[i]  <-  X[i-1] + lambda*(Mu-X[i-1])*dt + sigma*dw[i-1]
  }
  return(data.frame(t,X));
}

X3<-ornstein_uhlenbeck(0,2,Step = 0.001,Mu=2,lambda = 3,sigma = 2,X0=2)
ggplot(data = X3, aes(x=t, y=X)) +
    geom_line() +theme_bw()+
  ggtitle("Ornstein-Uhlenbeck process") +
  xlab("t") + ylab("X(t)")

X4<-ornstein_uhlenbeck(0,4,Step = 0.002,Mu=2,lambda = 3,sigma = 2,X0=2)
```

```r
ggplot(data = X4, aes(x=t, y=X)) +
    geom_line() +theme_bw()+
  ggtitle("Ornstein-Uhlenbeck process") +
  xlab("t") + ylab("X(t)")

### Quality of simulation
Distance<- max(abs(X4-X3))
Distance
ks_p_value<-NULL
K<-100
for (i in 1:K) {
  Yt1<-ornstein_uhlenbeck(0,2,Step = 0.001,Mu=2,lambda = 3,sigma = 2,X0=5)
  Yt2<-ornstein_uhlenbeck(0,4,Step = 0.002,Mu=2,lambda = 3,sigma = 2,X0=5)
  ks_p_value[i]<-ks.test(Yt1$X,Yt2$X)$p.value
}

plot(ks_p_value,type="o", xlab="Simulation",ylab="p-value",
     main="Plot of KS p-values")
points(rep(0.05,K),type = "l", col="red")

SDE_2D <- function(Begin, End,Step,X0){
  mu <- c(0,0) # Mean
  sigma <- matrix(c(Step, 0, 0, Step),2)
  t<-seq(Begin,End,Step) ### create sequence
  n=length(t)
  dt  <- Step
  X<- matrix(NA, nrow = n, ncol = 2)
  X[1,] <- X0
  for (i in 2:n) {
    drf1<-(t(c(1,-1,1))%*%matrix(c(X[(i-1), ],1)))*dt
    drf2<- ((t(c(-2,-1))%*%matrix(X[(i-1), ])))*dt
    Wt<- c(2*X[(i-1),1]-3,4)* mvrnorm(1, mu = mu, Sigma = sigma )
    X[i,]  <- c(X[(i-1), ]) + c(drf1,drf2) + Wt
  }
  X<-data.frame(X)
  colnames(X)<-c("X1t","X2t")
  return(data.frame(t,X))
}
X6<-SDE_2D(0,2,Step = 0.001,X0=c(1,-3))
X6%>% gather(key,value, X1t, X2t) %>%
    ggplot(aes(x=t, y=value, colour=key)) +
    geom_line() +theme_bw()+
  ggtitle("SDE") +
  xlab("t") + ylab("Value")
```

```r
ggplot(data = X6, aes(x=X1t, y=X2t)) +
    geom_point() +theme_bw()+
  ggtitle("SDE in 2-D") +
  xlab("X1(t)") + ylab("X2(t)")

peacock_p_value<-NULL
K<-100
for (i in 1:K) {
  Yt7<-SDE_2D(0,2,Step = 0.001,X0=c(1,-3))
  Yt8<-SDE_2D(0,4,Step = 0.002,X0=c(1,-3))
  peacock_p_value[i]<-peacock2(Yt8[,2:3],Yt7[,2:3])
}

plot(peacock_p_value,type="o", xlab="iteration",ylab="p-value",
     main="Plot of KS p-values")
points(rep(0.05,K),type = "l", col="red")


# this R markdown chunk generates a code appendix
```