Homework 8 - STAT 5361 Statistical Computing

Sen Yang[*]

03 November 2018

Abstract

This is homework 8 for STAT 5361 - Statistical Computing.

Contents

[*] sen.2.yang@uconn.edu; M.S. student at Department of Statistics, University of Connecticut.

1 Orstein–Uhlenbeck Process

1.1 Transition Distribution

SDE of Ornstein-Uhlenbeck process is

$$\mathrm{d}r(t) = \alpha(b - r(t))\mathrm{d}t + \sigma \mathrm{d}W(t)$$

Let $H(t) = \int_0^t -\alpha \mathrm{d}s = -\alpha t$. By Ito's formula,

$$\mathrm{d}(e^{\alpha t} r(t)) = e^{\alpha t}\alpha b \mathrm{d}t + e^{\alpha t}\sigma \mathrm{d}W(t)$$

That is,

$$\begin{aligned}
\int_0^t \mathrm{d}(e^{\alpha t} r(t)) &= \int_0^t e^{\alpha s}\alpha b \mathrm{d}s + \int_0^t e^{\alpha s}\sigma \mathrm{d}W(s) \\
e^{\alpha t} r(t) - e^0 r(0) &= \alpha b \int_0^t e^{\alpha s}\mathrm{d}s + \sigma \int_0^t e^{\alpha s}\mathrm{d}W(s) \\
r(t) &= e^{-\alpha t} r(0) + b(1 - e^{-\alpha t}) + e^{-\alpha t}\sigma \int_0^t e^{\alpha s}\mathrm{d}W(s) \\
e^{-\alpha\Delta} r(t) &= e^{-\alpha(t+\Delta)} r(0) + b(e^{-\alpha\Delta} - e^{-\alpha(t+\Delta)}) + e^{-\alpha(t+\Delta)}\sigma \int_0^t e^{\alpha s}\mathrm{d}W(s)
\end{aligned}$$

Since

$$r(t+\Delta) = e^{-\alpha(t+\Delta)} r(0) + b(1 - e^{-\alpha(t+\Delta)}) + e^{-\alpha(t+\Delta)}\sigma \int_0^{(t+\Delta)} e^{\alpha s}\mathrm{d}W(s),$$

Then,

$$\begin{aligned}
r(t+\Delta) &= e^{-\alpha\Delta} r(t) + b(1 - e^{-\alpha\Delta}) + e^{-\alpha(t+\Delta)}\sigma \int_t^{(t+\Delta)} e^{\alpha s}\mathrm{d}W(s) \\
&= e^{-\alpha\Delta} r(t) + b(1 - e^{-\alpha\Delta}) + e^{-\alpha(t+\Delta)}\sigma \sqrt{\int_t^{(t+\Delta)} e^{2\alpha s}\mathrm{d}s Z} \\
&= e^{-\alpha\Delta} r(t) + b(1 - e^{-\alpha\Delta}) + e^{-\alpha(t+\Delta)}\sigma \sqrt{\frac{e^{2\alpha(t+\Delta)} - e^{2\alpha}}{2\alpha}} Z \\
&= e^{-\alpha\Delta} r(t) + b(1 - e^{-2\alpha\Delta}) + \sigma \sqrt{\frac{1 - e^{-\alpha\Delta}}{2\alpha}} Z
\end{aligned}$$

1.2 A Random Walk for the Process

The algorithm to implement a random walk construction for the process:

Algorithm 1 Implement a random walk construction for the process

Step 1: Set $r(0) = 1, i = 1, \Delta = 1/500$.
Step 2: Sample $Z_i \sim N(0, 1)$.
Step 3: compute $r(i)$.
Step 4: $i = i + \Delta$.
if $i > \frac{500-1}{1/500} = 249500$ **then**
Break the loop.
else
Return Step 2.
end if

Realize this algorithm in R.

```
# Orstein-Uhlenbeck Process
alpha <- c(0.1,1,5)
sigma <- c(0.1,0.2,0.5)
b <- c(-5,5)

## random walk
ran_walk <- function(alpha,sigma,b,r0=1,T=500,delta=1/500) {
  dim<-length(alpha)*length(sigma)*length(b)
  results<-data.frame(matrix(0,(T-r0)/delta,dim*5))
  for (i in 1:dim) {results[,i*5-1]<-rnorm(nrow(results))}
  for (j in 1:length(alpha)) {
    for (k in 1:length(sigma)) {
      for (l in 1:length(b)) {
        num <- (j-1)*6+(k-1)*2+l
        results[,num*5-4]<-alpha[j]
        results[,num*5-3]<-sigma[k]
        results[,num*5-2]<-b[l]
      }
    }
  }
  ri<-data.frame(matrix(r0,1,dim))
  for (i in 1:nrow(results)) {
    ri<-exp(-results[i,(1:dim)*5-4]*delta)*ri[1,1:dim]+
      results[i,(1:dim)*5-2]*(1-exp(-2*results[i,(1:dim)*5-4]*delta))+
      results[i,(1:dim)*5-3]*sqrt((1-exp(-results[i,(1:dim)*5-4]*delta))/2/
             results[i,(1:dim)*5-4])*results[i,(1:dim)*5-1]
    results[i,(1:dim)*5]<-ri[1:dim]
  }
  results
}
results<-ran_walk(alpha,sigma,b)

## plot
res2<-results[,(1:dim)*5]
```

```
res2<-cbind(col(1:nrow(results)),res2)
library(ggplot)
for (j in 1:length(alpha)) {
  for (k in 1:length(sigma)) {
    for (l in 1:length(b)) {
      num <- (j-1)*6+(k-1)*2+l
      ggplot() + deom_line(data=res2,aes(x=res2[,1],y=res2[,num])) +
        labs(x="t",y="r(t)",
             title=expression(paste(alpha,"=",alpha[j],", ",sigma,"=",sigma[k],", b=",b[l]))) +
        theme(plot.title = element_text(hjust = 0.5))
    }
  }
}
```

1.3 Simulation of the Process by Euler–Maruyama method

2 Poisson Process

2.1 Distribution of $N(5)$

Given $T > 0$, $Z = \int_0^T \lambda(t)\mathrm{d}t = \int_0^T \sqrt{t} + e^{-t}\sin(2\pi t)\mathrm{d}t$. Calculate this integral by R,

```
## Integration
lamda_t <- function(t) {
  lamdat<- sqrt(t) + exp(-t)*sin(2*pi*t)
  lamdat
}

Z <- integrate(lamda_t,0,5)
Z$value
```

```
## [1] 7.607738
```

Therefore, $N(5) \sim \text{Poisson}(Z), where \;\; Z = 7.607738.$

2.2 Function to Simulate from this Poisson Process in R

Since $\lambda(t) = \sqrt{t} + e^{-t}\sin(2\pi t) \leq \sqrt{t} + e^{-t}$, we choose $\lambda_0(t) = \sqrt{t} + e^{-t}$. Therefore,

$$\Lambda(\tau) = \int_0^\tau \sqrt{t} + e^{-t}\mathrm{d}t = \frac{2}{3}\tau^{2/3} - e^{-tau} + 1$$

The algorithm to simulation from this Poisson process:

Algorithm 2 Simulation from a Poisson process

Step 1: Derive mean function $\Lambda(\tau)$ by $\Lambda(\tau) = \int_0^\tau \lambda(t)\mathrm{d}t$
Step 2: Generate $S_1, S_2, ...,$ from a homogeneous Poisson process with rate one
Step 3: Let $T_i = \Lambda^{-1}(S_i), i = 1, 2, ...$
Step 4: Sample $U \sim Unif(0, 1)$, for each i
if $U < \lambda(T_i)/\lambda_0(T_i)$ **then**
  Return T_i
else
  Go to Step 3.
end if

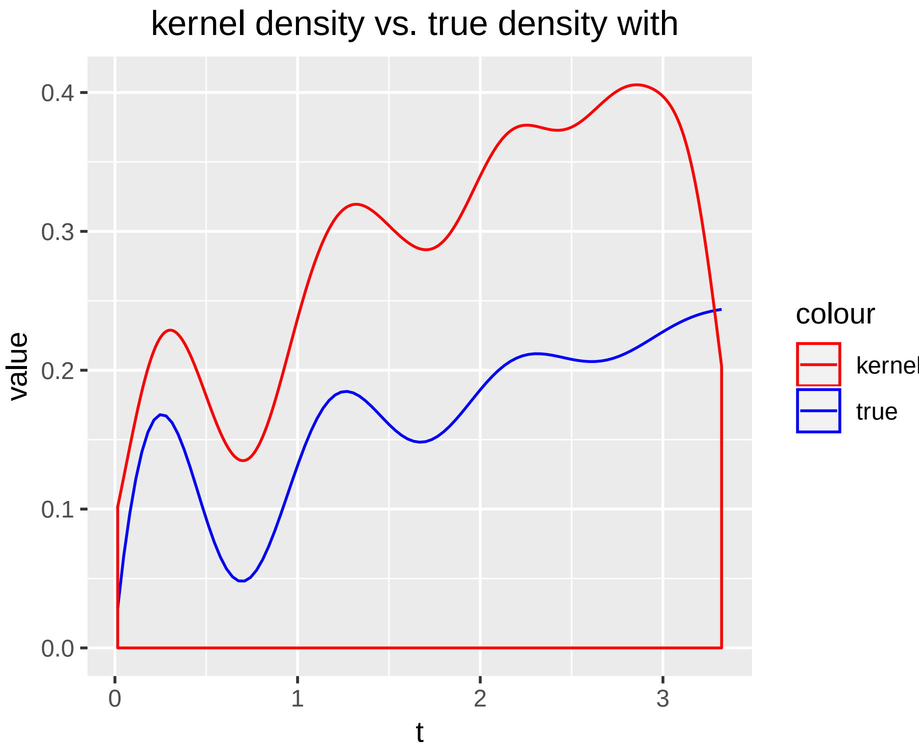
Build corresponding funcion in R.

```
## poisson process
tau_t<-matrix(0,1,1)
for (i in 1:1000) {
  S<-5* runif(rpois(1,5))
  T<-matrix(0,length(S),1)
for (j in 1: length(S)) {
  if (length(S)==0) next
  lamda_f <- function(t) {
    lamda_f <- 2/3*t^(3/2)-exp(-t)+1-S[j]
    lamda_f
  }
  T[j,1]<-uniroot(lamda_f,c(0,5))$root
  tau<-matrix(0,length(S),1)
}
  for (k in 1:nrow(T)) {
    if (length(T)==0) next
    u0 <- runif(1)
    if (u0<=(sqrt(T[k,1])+exp(-T[k,1])*sin(2*pi*T[k,1]))/(sqrt(T[k,1])+exp(-T[k,1]))) {
      tau[k,1]<-T[k,1]
    }
    else next
  }
  tau_t<-rbind(tau_t,tau)
}
tau_t<-as.data.frame(tau_t[tau_t>0 & tau_t<5])
```

2.3 Poisson Process Simulation

Generate events from this Poisson process 1000 times and plot the results.

```
true_f <- function(t) {
  true_f<- (sqrt(t)+exp(-t)*sin(2*pi*t))/(Z$value)
  true_f
}
```


kernel density vs. true density with
0.4
0.3
0.2
0.1
0.0
value
0
1
2
3
t
colour
kernel
true