Laboration: Simulation of Stochastic Processes

Simon Sigurdhsson

October 13, 2010

Abstract

Stochastic processes are a good way of describing many everyday processes dealing with arrivals and departures of various things: a queue at the supermarket, the number of cars on a bridge, and in some respects even our written language. It is therefore of interest to be able to simulate these processes.

1 Choice of language

For this assignment, I have been working with the statistical software *R*. It is closely related to S^+, and is, like S^+, an implementation of the *S* programming language developed at Bell Laboratories. *R* was chosen because of its ease of use, statistical packages, and availability (being both free *and* installed on the Chalmers remote servers). It is also very lightweight, compared to MATLAB, while still performing the tasks required in this assignment.

2 Vehicles on a bridge

Let us define a shot noise process $X(t)$, which can be thought of as a model of the number of vehicles on a bridge, when vehicles arrive with interarrival times exp(1)-distributed (thus arriving according to a Poisson process) and the bridge takes $\frac{1}{2}$ time units to cross:

$$g(t) = \begin{cases} 0, & t < 0 \\ 1, & 0 \leq t \leq \frac{1}{2} \\ 0, & \frac{1}{2} < t \end{cases}$$

Using *R*, we can simulate and plot $X(t)$, to get a better idea of how the process behaves:

```
g <- function(t) {
  ret <- as.numeric(t >= 0 & t <= 0.5)
  ret
}
xi <- rexp(50); xi <- xi[cumsum(xi) <= 10]
nu <- rexp(length(xi)); nu <- nu[cumsum(nu) <= 0.5]
t <- seq(0, 10, by=.01); X1 <- 0; X2 <- 0
for(k in 1:length(xi)) { X1 <- X1 + g(t-sum(xi[1:k])) }
for(k in 1:length(nu)) { X2 <- X2 + g(t+sum(nu[1:k])) }
```

```
if(length(nu) == 0) X2 <- as.numeric(!is.na(X2))
X <- X1 + X2
postscript("fig1a.eps")
plot(t, X); dev.off()
```

The result of this simple simulation can be seen in figure 1. As you can see, arrivals occur randomly and cause an “elevation” for $\frac{1}{2}$ time units. In this particular run, there are at most 2 cars on the bridge at any given time (i.e. $\max X(t) = 2$).

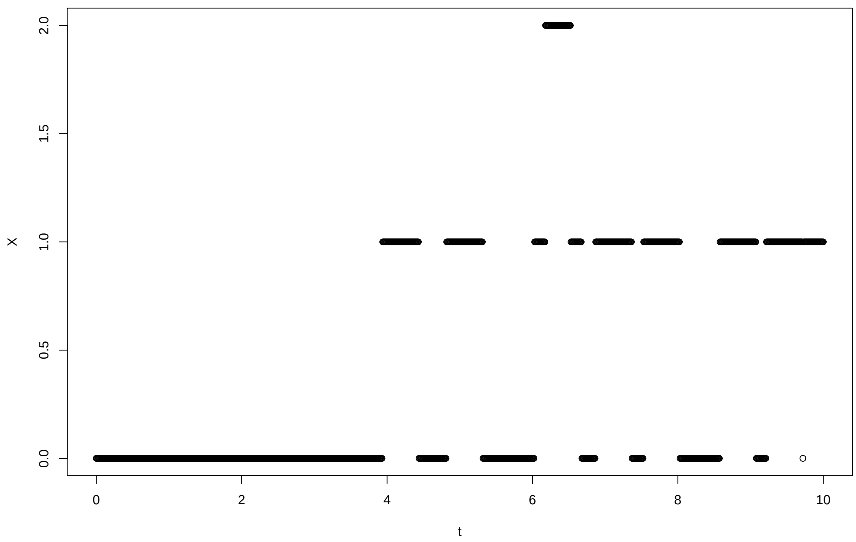


Figure 1: A simulation of the stochastic process $X(t)$

It may also be of interest to calculate the probability that there are, say, n or more vehicles on the bridge for some suitable value of n. To do this (here, we set $n = 3$) we can perform a large number of simulations (here, 10 000) accordning to the Monte Carlo principle, and then simply calculate the probability (with a given confidence interval, say 99%). To do this, we use the following R program:

```
p <- c()
for(i in 1:10000) {
  xi <- rexp(50); xi <- xi[cumsum(xi) <= 10]
  nu <- rexp(length(xi)); nu <- nu[cumsum(nu) <= 0.5]
  M <- c(); e <- FALSE;
  for(k in 1:length(nu)) { M <- c(M, -sum(nu[1:k])) }
  for(k in 1:length(xi)) { M <- c(M, sum(xi[1:k])) }
  M <- sort(M[-0.5 <= M & 10 >= M], TRUE)
  if(length(M)>=3) {
    for(a in 1:(length(M)-2)) {
      for(b in 2:(length(M)-a)) {
        if((M[a]-M[a+b]) <= 0.5) e <- TRUE
      }
    }
```

```
  }
  p <- if(e) c(p,1) else c(p,0)
}
pm <- mean(p)
cil <- qnorm(0.995)*sd(y)/sqrt(10000)
ci <- c(pm-cil, pm+cil)
```

The code above exploits the fact that, according to the Central Limit Theorem, p will be approximately $N(\mu, \frac{\sigma}{\sqrt{n}})$ as n gets large, where μ is the sample mean and σ the sample standard deviation. We can thus calculate the confidence interval as $\mu \pm \Phi^{-1}(1-\alpha/2)\frac{\sigma}{\sqrt{n}}$.

It also takes advantage of the fact that $X(t) \geq 3$ for some t if and only if there exists certain numbers $m_1, m_2, m_3 \in \mathbb{M}$ (where $\mathbb{M}$ is given by `M` in the above listing) such that

$$-\frac{1}{2} \leq m_1 < m_2 < m_3 \leq 10, \quad m_3 - m_1 \leq \frac{1}{2}.$$

Given this setup, we have a 99% confidence interval of 0.4984 ± 0.0129. What we in essence are doing is to simulate a random variable ζ, which is 1 if the given condition (i.e. $\max X(t) \geq 3$) holds and 0 otherwise, and calculate the expected value (sample mean) and a confidence interval with 10 000 simulated samples.

3 A stationary Gaussian process

Let's say we have a stationary Gaussian process given by

$$X(t) = \int_{-\infty}^{\infty} f(t+s)\mathrm{d}W(s),$$

$$f(t) = \begin{cases} 1-t^2, & |t| \leq 1 \\ 0, & |t| > 1 \end{cases}$$

where $W(s)$ is a Wiener process. It might be interesting to plot a simulation of this process, to see how it behaves. Though the integral may look like it's hard to compute, we're in luck — it can be approximated by a sum:

$$X(t) = \int_{-\infty}^{\infty} f(t+s)\mathrm{d}W(s) = \lim_{n\to\infty} \sum_{-s(n)}^{s(n)} f\left(t+\frac{k}{n}\right)\left(W\left(\frac{k+1}{n}\right) - W\left(\frac{k}{n}\right)\right)$$

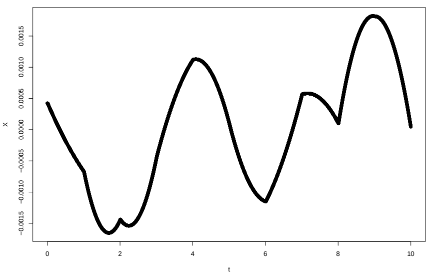
This reduces our problem significantly; we can now simply generate a suitable number of samples of the Wiener process, and apply the sum to these samples. To generate samples from the Wiener process, we first generate $2n+1$ samples with an $N(0, \frac{10}{n})$ distribution, and calculate the cumulative sum of these. This gives us a Wiener process for $t \in [-10, 10]$. In *R*, with $n = 10000$:

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

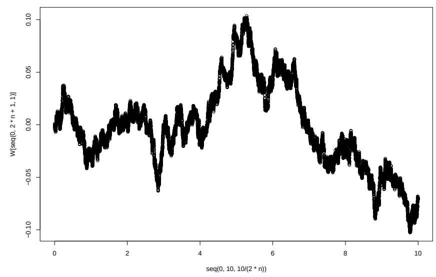
```
  r
}
n < - 10000
t < - seq(0, 10, 10/n)
N < - rnorm(2*n+1, 0, 10/n)
W < - cumsum(N)
sums < - seq(-n, n, 1)
X < - c()
for(k in t) {
  step < - sum(f(k+sums/sqrt(n))*(W[(sums+1)/sqrt(n)-sums[1]]
+                                  -W[sums/sqrt(n)-sums[1]]))
  X < - c(X, step)
}
postscript("fig2.eps")
plot(t, X); dev.off()
```

We set $s(n) = n^2$ (or rather, n inside the sum to $\sqrt{n}$) to satisfy the condition $\lim_{n\to\infty} \frac{s(n)}{n} = \infty$. The result can be seen in Figure 2a, and it's quite interesting. It is also interesting to compare the result to the plot of $1 - t^2$, (Figure 2c) and the underlying Wiener process (Figure 2b).

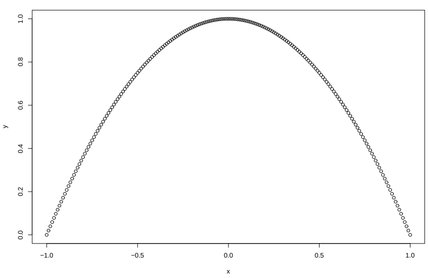
We can see that when the Wiener process grows very large (around $t = 5$) the stationary process tends to 0 — understandably, as f does the same thing. We can also see that the process reaches its largest values when the underlying Wiener process is close to 0 but slightly off. This is also to be expected, since f then is large, while the Wiener process itself provides a sign (and scales $X(t)$ down). As the figure shows, $X(t)$ lies between ± 0.0015, while $W(t)$ lies between ± 0.1.



(a) The Gaussian process $X(t)$



(b) The underlying Wiener process $W(t)$



(c) The function used, $f(t) = 1 - t^2$