Coursework - Stochastic Processes

CID: 02091191

March 22, 2022

1 First exercise

Consider the continuous time Markov process X_t on state space $E = \{1, 2\}$ with infinitesimal generator

$$\boldsymbol{G} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

(a) Simulation algorithm for this process

The main idea behind the simulation of continuous-time Markov processes is to use the alarm-clock formulation. To recall, we denote by H_i the holding time in state $i \in E$ as the further time until the chain changes its state

$$H_i = \inf\{s \geq 0 : X_{t+s} \neq i \mid X_t = i\} = \inf\{s \geq 0 : X_s \neq i \mid X_0 = i\}$$

As mentioned in the Lecture Notes [1], H_i is exponentially distributed with parameter $-g_{ii}$ where $g_{ij} = (\boldsymbol{G})_{ij}$. Therefore, after reaching state i, a series of independent alarm clocks are simultaneously set, whose alarm times are exponentially distributed, one for each other state. Note here that since there are only two distinct states, only one alarm needs to be set. Then, the first ringing alarm, that is the only that has been set here, defines the new state j and the process X_t moves from state i to state j. As mentioned, the general generation procedure simplifies here since we only have two different states, and a pseudo-code simulation algorithm of X_t for $t \in [0, T]$ is provided below.

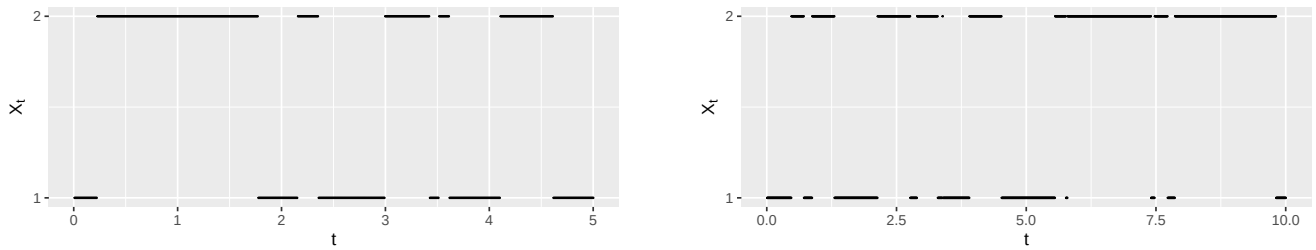
Algorithm 1 Simulation algorithm for the considered continuous-time Markov process

Require $\lambda > 0$, $\mu > 0$, $T \in (0, \infty)$, $X_0 \in \{1, 2\}$
$t \leftarrow 0$
$X \leftarrow$ empty list
while $t < T$ **do**
    Append last changing time and state (t, X_t) to X
    if current state is 1 **then** sample alarm $\delta \sim \text{Exp}(\lambda)$, new state is 2
    if current state is 2 **then** sample alarm $\delta \sim \text{Exp}(\mu)$, new state is 1
    $t \leftarrow \min\{t + \delta, T\}$
    $X_t \leftarrow$ new state
Return X

Using Algorithm 1, one can then obtain a table of values containing the changing times and states.

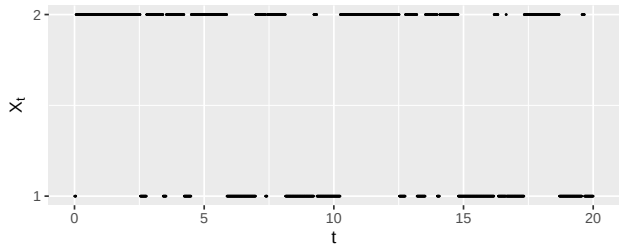
(b) Implementation and generations of the process

Consider $\lambda = 2$ and $\mu = 1$. One can use the Algorithm 1 to generate simulations of the defined process. Then, one can build a function that retrieves the *continuous-time* form of the process X_t (at least with a small time step) from the changing points sequence generated by Algorithm 1. Such realisations are presented on Figure 1.1.

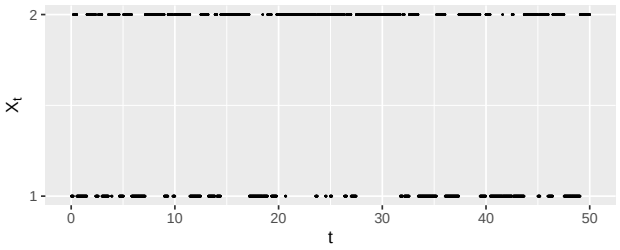


(a) $T = 5$

(b) $T = 10$



(c) $T = 20$



(d) $T = 50$

Figure 1.1: Realisations of the continuous-time Markov process for different values of T, $X_0 = 1$

As shown on Figure 1.1, the time spent by X_t in each space $\{1, 2\}$ is different: indeed, X_t seems to be more often in state 2 than in state 1. To assess this quantitatively, one can derive the theoretical invariant distribution $\boldsymbol{\pi} = (\pi_1 \ \pi_2)$ which satisfies $\boldsymbol{\pi} = \boldsymbol{\pi P_t}$ where $\boldsymbol{P_t}$ is the transition function of the Markov process. In fact, it can be shown that the stationary distribution $\boldsymbol{\pi}$ satisfies $\boldsymbol{\pi G} = \mathbf{0}$. Solving this equation therefore derives the invariant distribution:

$$\begin{cases} \boldsymbol{\pi G} = \mathbf{0} \\ \pi_1 + \pi_2 = 1 \end{cases} \Leftrightarrow \begin{cases} -\lambda\pi_1 + \mu\pi_2 = 0 \\ \pi_1 + \pi_2 = 1 \end{cases}$$
$$\Leftrightarrow \begin{cases} \pi_1 = \frac{\mu}{\lambda+\mu} = \frac{1}{3} \\ \pi_2 = \frac{\lambda}{\lambda+\mu} = \frac{2}{3} \end{cases}$$

Therefore, as $T \to \infty$, the proportion of samples in state 1 should be $\pi_1 = \frac{1}{3}$ and the proportion of samples in state 2 should be $\pi_1 = \frac{2}{3}$, which is consistent with Figure 1.1 where we observe more samples in state 2 than in state 1 especially for $T = 50$. To assess this using simulation, one can simulate a trajectory X_t for $t \in [0, T]$ with a high value of T and plot the proportion of samples in each state on $[0, t]$.

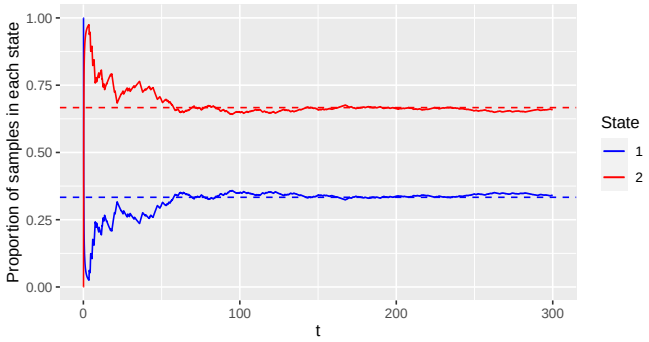


Figure 1.2: Distribution of samples within each state as a function of time

Figure 1.2 shows the evolution of proportion of samples in each state from $t = 0$ to t where t varies in $[0, T]$ with $T = 300$. The dashed lines represent the invariant distribution derived previously, and therefore one may assess that the long-time simulations distribution converge to the theoretical invariant distribution. In particular here, at $t = T = 300$, the proportion of X values in state 1 is 0.3399 ($\simeq \frac{1}{3}$) and the proportion of X values in state 2 is 0.6601 ($\simeq \frac{2}{3}$). This ensures validity of the generating algorithm.

(c) Inference on μ

Assume now one makes observations of the output from a single run of the chain with infinitesimal generator $\boldsymbol{G}$ with $\lambda = 2$ and μ unknown at times $t = \Delta, 2\Delta, ...$ for $\Delta > 0$. Let K_{ij} be the number of transitions from state i to state j i.e. the number of times $X_{n\Delta} = i$ and $X_{(n+1)\Delta} = j$ for some n. Note that $X_{\Delta k} = i_k \in \{1, 2\}$ for $k = 1, ..., n$. The likelihood of observing the transitions is

$$\begin{aligned}
L(\mu) &= \mathbb{P}[X_{n\Delta}, X_{(n-1)\Delta}, ..., X_\Delta | \mu, \lambda, X_\Delta] \\
&= \mathbb{P}[X_{n\Delta} = i_n, X_{(n-1)\Delta} = i_{n-1}, ..., X_\Delta = i_1 | X_\Delta = i_1] \\
&= \mathbb{P}[X_{n\Delta} = i_n | X_{(n-1)\Delta} = i_{n-1}, ..., X_\Delta = i_1] \cdot \mathbb{P}[X_{(n-1)\Delta} = i_{n-1}, ..., X_\Delta = i_1] \\
&= \mathbb{P}[X_{n\Delta} = i_n | X_{(n-1)\Delta} = i_{n-1}] \cdot \mathbb{P}[X_{(n-1)\Delta} = i_{n-1}, ..., X_\Delta = i_1] \\
&= \mathbb{P}[X_{n\Delta} = i_n | X_{(n-1)\Delta} = i_{n-1}] \cdot ... \cdot \mathbb{P}[X_{2\Delta} = i_2 | X_\Delta = i_1] \\
&= \mathbb{P}[X_\Delta = i_n | X_0 = i_{n-1}] \cdot ... \cdot \mathbb{P}[X_\Delta = i_2 | X_0 = i_1] \\
&= \prod_{i=1}^{2} \prod_{j=1}^{2} \mathbb{P}[X_\Delta = i | X_0 = j]^{K_{ij}} \\
L(\mu) &= \prod_{i=1}^{2} \prod_{j=1}^{2} P(\Delta)_{ij}^{K_{ij}}
\end{aligned}$$

where $P(\Delta)_{ij} = \mathbb{P}[X_\Delta = i | X_0 = j]$ is the transition function of the continuous-time Markov process. This derivation uses the chain rule for conditional probabilities with more than 2 events on line 3, iteratively until we obtain the result on line 5. The Markov property of the process is also used on line 4. Finally, line 6 is obtained after shifting time.

Using the backward Kolmogorov equation $\boldsymbol{P}_t' = \boldsymbol{G}\boldsymbol{P}_t$, one can show that

$$\begin{aligned}
P_{11}'(t) &= -P_{11}(t)g_{11} + P_{12}(t)g_{21} \\
&= -\lambda P_{11}(t) + (1 - P_{11}(t))\mu \\
&= \mu - (\lambda + \mu)P_{11}(t)
\end{aligned}$$

The solution of the linear differential equation is

$$P_{11}(t) = \frac{\mu}{\lambda + \mu} + \frac{\mu}{\lambda + \mu} e^{-(\lambda+\mu)t}$$

Similarly,

$$P_{22}(t) = \frac{\lambda}{\lambda + \mu} + \frac{\mu}{\lambda + \mu} e^{-(\lambda+\mu)t}$$

And

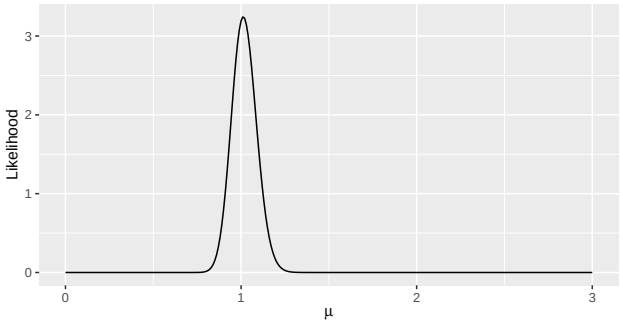
$$\begin{aligned}
P_{12}'(t) &= P_{11}(t)g_{12} + P_{12}(t)g_{22} \\
&= P_{11}(t)\lambda - \mu P_{12}(t) \\
&= \lambda - (\lambda + \mu)P_{11}(t)
\end{aligned}$$

which can be solved similarly as well as $P_{21}(t)$. The final resulting transition function is

$$\boldsymbol{P}(t) = \frac{1}{\lambda + \mu} \begin{pmatrix} \mu + \lambda e^{-(\lambda+\mu)t} & \lambda - \lambda e^{-(\lambda+\mu)t} \\ \mu - \mu e^{-(\lambda+\mu)t} & \lambda + \mu e^{-(\lambda+\mu)t} \end{pmatrix}$$

(d) Maximising the likelihood

Given a time period Δ, one can compute matrix K. Now that the likelihood has been defined as a function of μ, one can easily use a generated trajectory to infer the parameter μ.



(a) Likelihood

(b) Log-likelihood

Figure 1.3: Plots of the likelihood and log-likelihood as functions of μ given the output of (b). For convenience, the likelhood values have been multiplied by a simple factor to re-scale the y-axis.

Figure 1.3 shows plots of the likelihood and log-likelihood as functions of μ. Since the likelihood values are very close to zero (even more when $\Delta \to 0$ since K values increase), one could also take into account the log-likelihood function in order to avoid numerical errors. These plots are consistent since both likelihood and log-likelihood functions seem to be maximised when $\mu \simeq 1$. Using the `optimize` function in `R` from the `stats` package, one could find that the likelihood and log-likelihood are both maximised for $\mu = 1.0125$, which is therefore the maximum likelihood estimator of μ on this generated trajectory and is close to the real value used to generate the trajectory which was 1.

The `R` code used for this question can be found below.

```
# Exercise 1
library(ggplot2)
library(scales)

# (a)
T <- 1000
lambda <- 2
mu <- 1
G <- matrix(c(-lambda, lambda, mu, -mu), nrow=2, ncol=2, byrow=TRUE)

run.simulation <- function(state, T, G) {
  trajectory <- matrix(nrow = 0, ncol = 2)
  t <- 0

  while (t < T) {

    cat(t, ": ", state, "\n")
    trajectory <- rbind(trajectory, c(t, state))

    #identify the other two states different from the current
    other.index <- (state) %% 2 + 1

    alarm <- -log(runif(1))/G[state,other.index]

    t <- min(T, t + alarm)
    state <- other.index
  }

  return(trajectory)
}

state <- 1
trajectory <- run.simulation(state, T, G)
```

```r
delta <- 1
eval.points <- seq(0, T, delta)

get.state <- function(x, trajectory) {
  lt <- which(trajectory[,1] < x)
  if (length(lt) > 0) {
    ind <- max(lt)
    return(trajectory[ind, 2])
  } else {
    return(NULL)
  }
}

states <- unlist(sapply(eval.points, function(x) get.state(x, trajectory) ))
eval.points <- eval.points[-1]

ggplot() +
  geom_point(aes(x=eval.points, y=states), size=0.1) +
  xlab("t") + ylab(expression(X[t])) +
  scale_y_continuous(breaks=c(1,2))

# (b)
proportion <- c()
for(i in 1:length(states)){
  states_to_now <- states[1:i]
  prop_1 <- mean(states_to_now == 1)
  prop_2 <- mean(states_to_now == 2)
  proportion <- rbind(proportion, c(prop_1, prop_2))
}

ggplot() +
  geom_line(aes(x=eval.points, y=proportion[,1], colour="Proportion of 1")) +
  geom_line(aes(x=eval.points, y=proportion[,2], colour="Proportion of 2")) +
  geom_hline(yintercept=1/3, colour="blue", linetype="dashed") +
  geom_hline(yintercept=2/3, colour="red", linetype="dashed") +
  xlab("t") + ylab("Proportion of samples in each state") +
  scale_colour_manual(name = 'State',
                      values =c('Proportion of 1'='blue','Proportion of 2'='red'),
                      labels = c('1','2'))

mean(states == 1)
mean(states == 2)

# (c) and (d)
generate.K <- function(states){
  K <- matrix(0, ncol=2, nrow=2)
  for(i in 1:(length(states)-1)){
    if(states[i] == 1 & states[i+1] == 1) K[1, 1] <- K[1, 1] + 1
    if(states[i] == 1 & states[i+1] == 2) K[1, 2] <- K[1, 2] + 1
    if(states[i] == 2 & states[i+1] == 1) K[2, 1] <- K[2, 1] + 1
    if(states[i] == 2 & states[i+1] == 2) K[2, 2] <- K[2, 2] + 1
  }
  return(K)
}

K <- generate.K(states)

likelihood <- function(mu){
  lambda <- 2
  P <- matrix(c(mu + lambda * exp(-(lambda + mu) * delta),
                lambda - lambda * exp(-(lambda + mu) * delta),
```

```
              mu - mu * exp(-(lambda + mu) * delta),
              lambda + mu * exp(-(lambda + mu) * delta)), ncol=2, nrow=2, byrow=TRUE)
  P <- 1/(lambda + mu) * P
  L <- 1
  for(i in 1:2){
    for(j in 1:2){
      L <- L * P[i, j]^(K[i, j])
    }
  }
  return(L)
}

log.likelihood <- function(mu){
  lambda <- 2
  P <- matrix(c(mu + lambda * exp(-(lambda + mu) * delta),
              lambda - lambda * exp(-(lambda + mu) * delta),
              mu - mu * exp(-(lambda + mu) * delta),
              lambda + mu * exp(-(lambda + mu) * delta)), ncol=2, nrow=2, byrow=TRUE)
  P <- 1/(lambda + mu) * P
  L <- 0
  for(i in 1:2){
    for(j in 1:2){
      L <- L + K[i, j] * log(P[i, j])
    }
  }
  return(L)
}

x <- seq(0, 3, 0.01)
y <- sapply(X=x, FUN=likelihood)
ggplot() +
  geom_line(aes(x=x, y=y*1e277)) +
  xlab(expression(mu)) + ylab("Likelihood")

x <- seq(0, 3, 0.01)
y <- sapply(X=x, FUN=log.likelihood)
ggplot() +
  geom_line(aes(x=x, y=y)) +
  xlab(expression(mu)) + ylab("Log-likelihood")

optimize(likelihood, interval=c(0, 2), maximum = TRUE)
optimize(log.likelihood, interval=c(0, 2), maximum = TRUE)
```

2 Second exercise

Consider the following continuous-time stochastic process, known as the Zig-Zag process defined as follows.

- $\frac{d}{dt}X_t = V_t$, where $X_0 = 0$
- $V_t = (-1)^{N_t}V_0$
- N_t is a non-homogeneous Poisson process with intensity function $\lambda(t) = \max(0, V_t X_t)$

(a) Generating the Zig-Zag process

The sampling process presented here has been extracted from [3]. To generate a Zig-Zag process trajectory, it is sufficient to simulate the switching times $(T_i)_{i\in\mathbb{N}}$. Given initial conditions (X_0, V_0) and switching times $(T_i)_{i\in\mathbb{N}}$, we denote by $Z_t = (X_t, V_t)$ the position-velocity process defined for all $t \geq 0$ as

$$\begin{aligned} V_t &= (-1)^k V_0,\ t \in [T_k, T_{k+1}) \\ X_t &= X_{T_k} + (t - T_k)V_{T_k},\ t \in [T_k, T_{k+1}) \end{aligned}$$

Therefore, given the state $(X_{T_0}, V_{T_0}) = (X_0, V_0)$ at switching time T_0, the next random switching time is given by $T_1 = T_0 + \tau$ where τ satisfies

$$\mathbb{P}[\tau > t] = \exp\left(-\int_0^t \lambda(X_0 + sV_0)ds\right)$$

In `R`, we can easily compute numerically $G(t) = \int_0^t \lambda(X_0 + sV_0)ds$ and its inverse

$$H(y) = \inf\{t \geq 0 : G(t) \geq y\}, y \in [0, 1]$$

Then, the random variable $\tau = H(-\log u)$ where $u \sim \mathcal{U}([0, 1])$. The full Zig-Zag process sampling is described in Algorithm 2.

Algorithm 2 Simulation algorithm of the Zig-Zag process

Require λ intensity function, (X, V) initial condition, $T \in (0, \infty)$
$(T_0, X_{T_0}, V_{T_0}) \leftarrow (0, X_0, V_0)$
$k \leftarrow 0$
while $T_k < T$ **do**
    Draw $u \sim \mathcal{U}([0, 1])$
    $\tau \leftarrow H(-\log u)$
    $T_{k+1} \leftarrow \max(T_k + \tau, T)$
    $X_{T_{k+1}} \leftarrow X_{T_k} + \tau V_{T_k}$
    $V_{T_{k+1}} \leftarrow -V_{T_k}$
    $k \leftarrow k + 1$
Return $(T_k, X_{T_k}, V_{T_k})_{k=0}^{\infty}$

(b) Realisations of the process

Plots illustrating 10 different realisations of the Zig-Zag process with $T = 100$, $V_0 \in \{1, 2, 3, 5, 10\}$ and $X_0 = 0$ are presented in Figure 2.1. One can see that V_0 plays an important role in the evolution of the process when T is fixed: increasing the velocity makes the process change more often its direction and increase the number of changing points when T is fixed. One could therefore see a similar behaviour when fixing V_0 and increasing T: the number of changing points would increase but the period between each changing point would however remain globally the same.

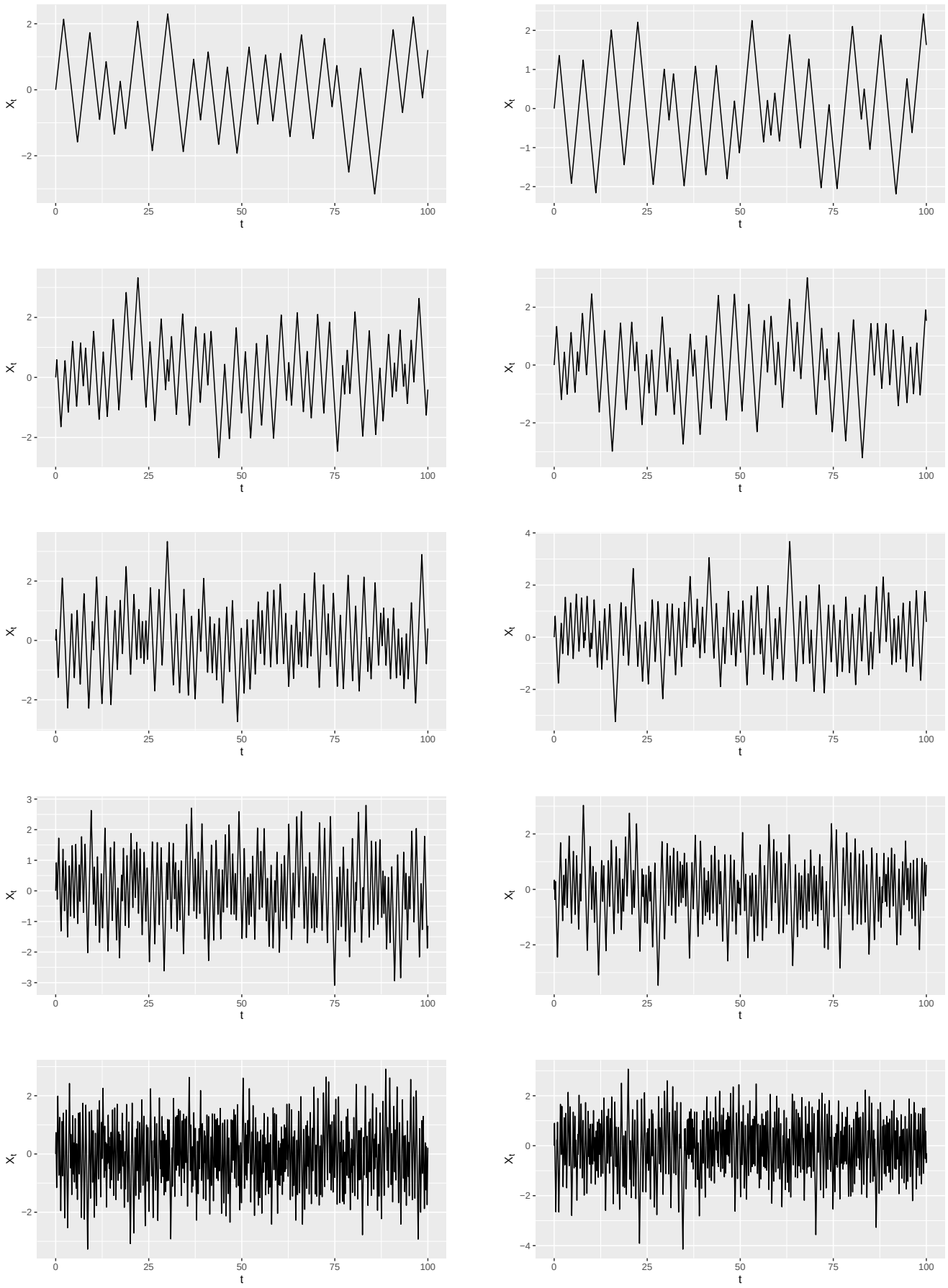
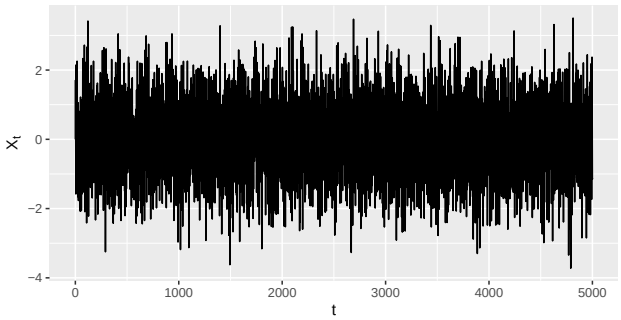


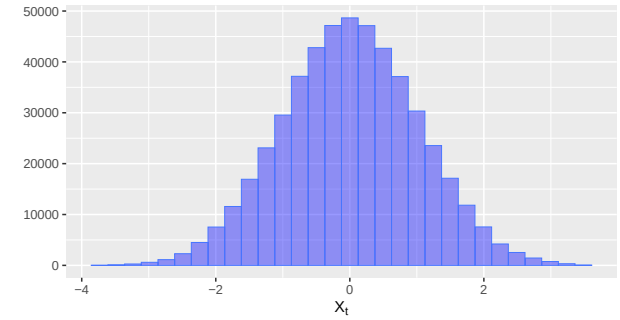
Figure 2.1: Examples of realisations of the Zig-Zag stochastic process with $T = 100$. From top line to bottom line, $V_0 \in \{1, 2, 3, 5, 10\}$.

(c) Long-time distribution of the process

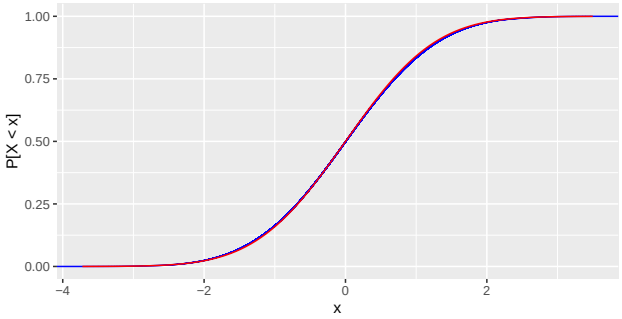
Now that the Zig-Zag process can be generated, one could focus on the long-time or stationary distribution of the process. Since Algorithm 2 provides a procedure to generate the changing points of the process, one could create a function in `R` which reconstructs the process X_t for all $t \in [0, T]$ and not only at the changing points. Studying the long-time distribution of the process could then consist in plotting the distribution of X_t values on $[0, T]$ when T is sufficiently high for the distribution to converge. Here, we assume that the long-time distribution of X_t is Gaussian. Therefore, we generated a Zig-Zag process with $X_0 = 0$, $V_0 = 1$ and $T = 5 \cdot 10^3$. This trajectory is then evaluated at time $0, \Delta, 2\Delta, ...$ with a time period $\Delta = 10^{-2}$.



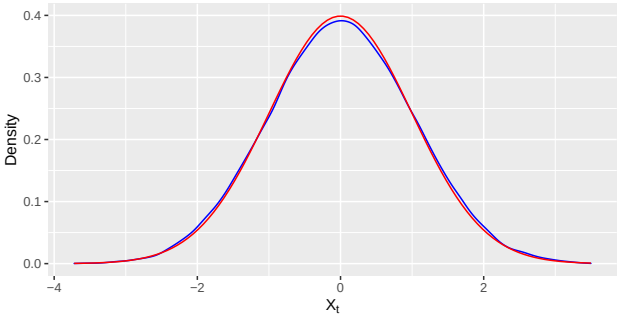
(a) Trajectory of the Zig-Zag process considered.



(b) Histogram of X_t values at $t = 0, \Delta, 2\Delta, ...$



(c) Empirical CDF (blue) v.s. $\mathcal{N}(0, 1)$ CDF (red)



(d) Kernel density (blue) v.s. $\mathcal{N}(0, 1)$ density (red)

Figure 2.2: Summary plots of the generated long-time Zig-Zag process. Comparison with a $\mathcal{N}(0, 1)$ distribution.

Figure 2.2 shows the trajectory of the process which contains a lot of changing points. The histogram of X_t values seems to follow a normal distribution: one could therefore compute the empirical mean and standard deviation of X_t at times $t = 0, \Delta, 2\Delta, ...$ as

$$\mu = \frac{\Delta}{T} \sum_{k=0}^{T/\Delta} X_{k\Delta}, \quad \sigma^2 = \frac{\Delta}{T} \sum_{k=0}^{T/\Delta} (X_{k\Delta} - \mu)^2$$

In `R`, we numerically estimate $\mu = 0.0065$ and $\sigma = 1.0177$. These values are very close to 0 and 1: therefore, the long-time distribution of X_t may follow a standard normal distribution. On Figure 2.2, we compare the long-time distribution with a $\mathcal{N}(0, 1)$. Both CDF and PDF seem very close to the $\mathcal{N}(0, 1)$ distribution. Proceeding to a Kolmogorov-Smirnov test in `R` returns a D statistic of 0.0271 and an associated p-value of 0.4561, which would show evidence that X_t follows a standard normal distribution. Hence, one may conclude that the long time distribution of X_t is a standard normal distribution.

Furthermore, as shown in [3], the moments of the Zig-Zag process can be computed more precisely than using a numerical integration as previously. The idea here is to note that X_t moves linearly between switches and thus can be decomposed into a sum of straight lines. Thus, the mean of X_t at time $T = T_K$

can be calculated as

$$\pi_{T_K}(f) = \frac{1}{T_K}\sum_{k=0}^{K-1}\int_{X_{T_k}}^{X_{T_{k+1}}} f(x)dx = \frac{1}{\sum_{k=0}^{K-1}\tau_k}\sum_{k=0}^{K-1}\int_0^{\tau_k} f\left(X_{T_k} + V_{T_k}s\right)ds$$

where $\tau_k = T_{k+1} - T_k$. Hence, in particular, the first and second moment of X_t take the form

$$\pi_{T_K}(x) = \frac{1}{\sum_{k=0}^{K-1}\tau_k}\sum_{k=0}^{K-1}\tau_k X_{T_k} + \frac{1}{2}V_{T_k}\tau_k^2$$

and

$$\pi_{T_K}(x^2) = \frac{1}{\sum_{k=0}^{K-1}\tau_k}\sum_{k=0}^{K-1}V_{T_k}\frac{-X_{T_k}^3 + (\tau_k V_{T_k} + X_{T_k})^2}{3}$$

In R, computing $\mu = \pi_T(x)$ and $\sigma = \sqrt{\pi_T(x^2) - \mu^2}$ gives the following estimates: $\mu = 0.0065$ and $\sigma = 1.0358$ which are equal or very close to the ones obtained using numerical integration. Thus, assuming the long-time distribution of X_t is Gaussian, it enforces the idea that it follows a standard normal distribution.

One may think that the long-time distribution depends on the intensity function chosen for the Poisson process N_t. However, a surprising result arises when doing the same procedure with different values of V_0. In fact, doing the same study with a different value of V_0 leads to the same long-time distribution parameters. This is due to the equivalence between increasing V_0 when T is fixed and increasing T when V_0 is fixed.

As the conducted work has been done only on a single trajectory, one could try to repeat it a higher number of times on multiple different trajectories in order to look at the fluctuations in mean and variance of the long time distribution. Therefore, we simulated 10^3 trajectories of Zig-Zag process with $T = 100$ (lowered for complexity reason).

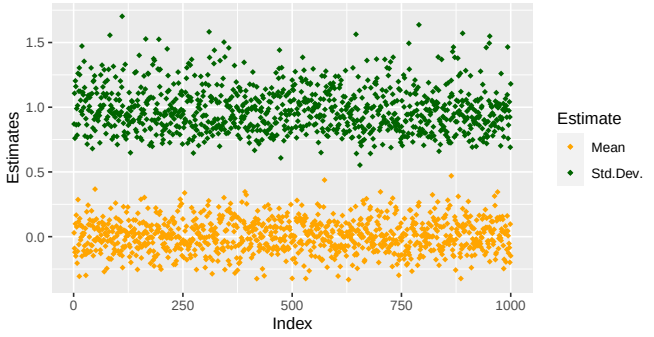


Figure 2.3: Estimates using exact integration of the mean and standard deviation of X_t among 10^3 sampled trajectories

As shown on Figure 2.3, the mean over each iteration of the process is around 0 and the standard deviation is around 1. Increasing T would give more accurate estimates. The averaged on all trajectories mean is 0.0074 and standard deviation is 0.9847. The standard error on the averaged mean (computed as the empirical standard deviation of all means) is 0.1269 and the one of standard deviation is 0.1752. Thus, the estimates of the standard deviation are slightly more sparse than the ones of the mean.

The R code used for this question can be found below.

```
# Exercise 2
library(ggplot2)

# (a)
```

```r
# R function which returns the inverse of a function f in [lower, upper]
invert <- function(f, lower, upper){
  function (y) uniroot((function (x) f(x) - y), lower=lower,
                       upper=upper, tol=1e-5)[1]
}

zigzag <- function(T, X0=0, V0=1){

  # Initialization
  X <- X0
  V <- V0
  tau <- 0
  k <- 1

  while (tau[length(tau)] < T){
    k <- k + 1

    lambda <- function(s) max(0, (X[k-1] + s * V[k-1]) * V[k-1])
    G <- function(t) integrate(f=Vectorize(lambda), lower=0, upper=t,
                               abs.tol = 1e-9)$value

    min_log_u = - log(runif(1))
    H <- invert(function(t) G(t), 0, 20) # It's enough to find an inverse on this interval

    # Set new changing time
    tau.k <- tau[k-1] + H(min_log_u)$root
    if (tau.k > T) tau.k <- T # If it goes further than T, then just put T
    tau <- c(tau, tau.k)

    # Velocity changes sign
    V_k <- - V[k-1]
    V <- c(V, V_k)
    # New position is updated
    X_k <- X[k-1] + (tau[k] - tau[k-1]) * V[k-1]
    X <- c(X, X_k)
  }
  return(list(tau=tau, X=X, V=V))
}

# (b)
trajectory <- zigzag(1e3)

ggplot() +
  geom_line(aes(x=trajectory$tau, y=trajectory$X)) +
  xlab("t") + ylab(expression(X[t]))

# (c)
# Interpolate with delta time
find.previous.tau <- function(t){
  i <- 1
  last.tau <- trajectory$tau[i]
  while(t > last.tau){
    i <- i + 1
    last.tau <- trajectory$tau[i]
  }
  last.tau <- trajectory$tau[i-1]
  last.vel <- trajectory$V[i-1]
  last.pos <- trajectory$X[i-1]
  return(c(last.tau, last.vel, last.pos))
}
```

```
times <- seq(trajectory$tau[1], trajectory$tau[length(trajectory$tau)], by=0.01)
X_t <- rep(0, length(times))

for(i in 2:length(times)){
  last.tau.vel <- find.previous.tau(times[i])
  last.tau <- last.tau.vel[1]
  last.vel <- last.tau.vel[2]
  last.pos <- last.tau.vel[3]
  print(i)
  X_t[i] <- last.pos + (times[i] - last.tau) * last.vel
}

ggplot() +
  geom_histogram(aes(x=X_t), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) + ylab("") + xlab(expression(X[t]))

ggplot() +
  stat_ecdf(aes(x=X_t), colour="blue") +
  geom_function(fun=pnorm, colour="red") +
  xlab("x") + ylab("P[X < x]")

ggplot() +
  geom_density(aes(x=X_t), colour="blue") +
  geom_function(fun=dnorm, colour="red") +
  xlab(expression(X[t])) + ylab("Density")

mean(X_t)
sd(X_t)

K <- length(trajectory$tau)
s <- 0
for(k in 1:(K-1)){
  T <- trajectory$tau
  X <- trajectory$X
  V <- trajectory$V
  tau <- T[k+1] - T[k]
  s <- s + tau * X[k] + 0.5 * V[k] * tau^2
}
s <- s/T[length(T)]
s

sd <- 0
for(k in 1:(K-1)){
  T <- trajectory$tau
  X <- trajectory$X
  V <- trajectory$V
  tau <- T[k+1] - T[k]
  sd <- sd + V[k] * (-X[k]^(1+2) + (tau * V[k] + X[k])^(1+2))/(2+1)
}
sd <- sd/T[length(T)]
sqrt(sd^2 - s^2)


mean_list <- c()
sd_list <- c()

for(iter in 1:1e3){
  trajectory = zigzag(T=1e2, X0=0, V0=1)
  K <- length(trajectory$tau)
  s <- 0
  for(k in 1:(K-1)){
```

```
    T <- trajectory$tau
    X <- trajectory$X
    V <- trajectory$V
    tau <- T[k+1] - T[k]
    s <- s + tau * X[k] + 0.5 * V[k] * tau^2
  }
  s <- s/T[length(T)]
  s

  sd <- 0
  for(k in 1:(K-1)){
    T <- trajectory$tau
    X <- trajectory$X
    V <- trajectory$V
    tau <- T[k+1] - T[k]
    sd <- sd + V[k] * (-X[k]^(1+2) + (tau * V[k] + X[k])^(1+2))/(2+1)
  }
  sd <- sd/T[length(T)]
  sd
  mean_list <- c(mean_list, s)
  sd_list <- c(sd_list, sqrt(sd^2 - s^2))
  print(iter)
}

mean(mean_list)
sd(mean_list)
mean(sd_list)
sd(sd_list)

ggplot() +
  geom_point(aes(x=1:1000, y=mean_list, colour="Mean"), shape=18) +
  geom_point(aes(x=1:1000, y=sd_list, colour="Std. Dev."), shape=18) +
  ylab("Estimates") + xlab("Index") +
  scale_colour_manual(name = 'Estimate',
                      values =c('Mean'='orange','Std. Dev.'='darkgreen'),
                      labels = c('Mean','Std.Dev.'))
```

3 Third exercise

Let X_t denote the standard Brownian Bridge on $[0,1]$ and let $M = \max_{t\in[0,1]} X_t$.

(a) Simulate the distribution of M

To simulate multiple realisations of M, one could generate multiple realisations of X_t and compute its maximum. Generating X_t may be done generating a standard Brownian Motion B_t with $t \in [0,1]$ and use the property that $B_t - tB_1$ is a Brownian Bridge.

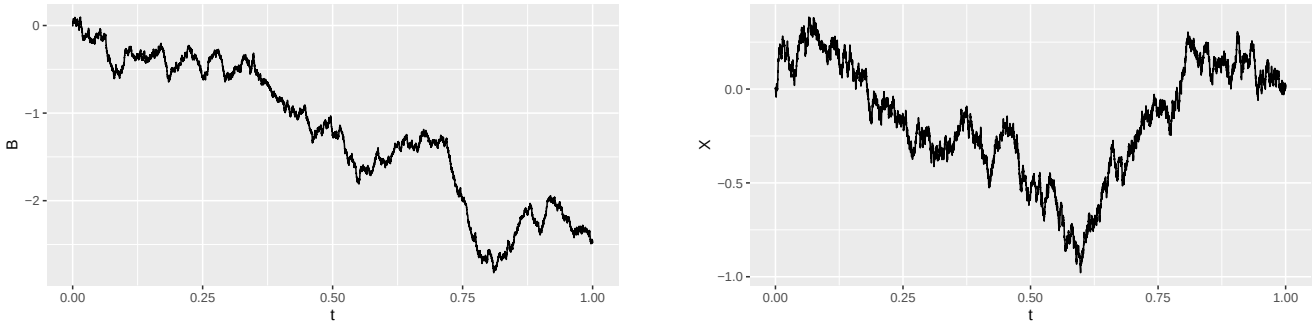
Therefore, one should first be able to generate a standard Brownian Motion: the pseudo-code for this generating procedure is shown in Algorithm 3. Then, this algorithm could be used to sample standard Brownian Bridges as shown on Algorithm 4.

Algorithm 3 Generate a Brownian Motion B_t, $t \in [0,1]$

1: **Require** n, the number of points to be considered within $[0,1]$
2: $t \leftarrow 0,\ B_0 \leftarrow 0$
3: $\Delta \leftarrow \frac{1}{n}$
4: **Repeat** $n-1$ times
5: $\quad B_{t+\Delta} \leftarrow B_t + \mathcal{N}(0,\Delta)$
6: $\quad t \leftarrow t + \Delta$
7: **Return** B_t

Algorithm 4 Generate a Brownian Bridge X_t, $t \in [0,1]$

1: **Require** n, the number of points to be considered within $[0,1]$
2: Generate a standard Brownian Motion B_t using n points
3: $X_t \leftarrow B_t - tB_1$
4: **Return** X_t



(a) Brownian Motion on $[0,1]$

(b) Brownian Bridge on $[0,1]$

Figure 3.1: Examples of standard Brownian Motion and Brownian Bridges generated using Algorithms 3 and 4 with $n = 10^4$. Note that the Brownian Bridge shown is not the Brownian Bridge associated to the Brownian Motion, but these are two separate examples.

Figure 3.1 displays an example of each considered stochastic process generated using the given pseudo-codes. Now that the generating procedure for a standard Brownian Bridge is described, one could easily sample multiple standard Brownian Bridges on $[0,1]$ and compute their maximum reached value M. Thus, sampling K Brownian Bridges would lead to obtaining K different M values. Using $n = 10^4$ and $K = 10^4$, the histogram of M values is shown on Figure 3.2.

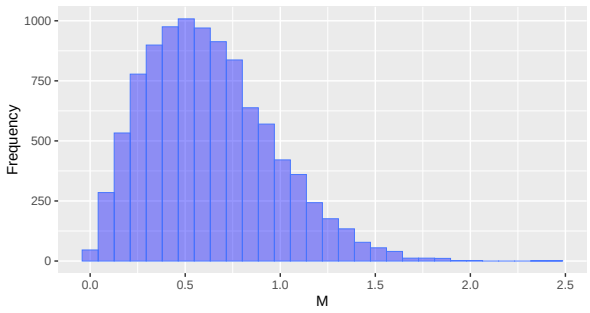
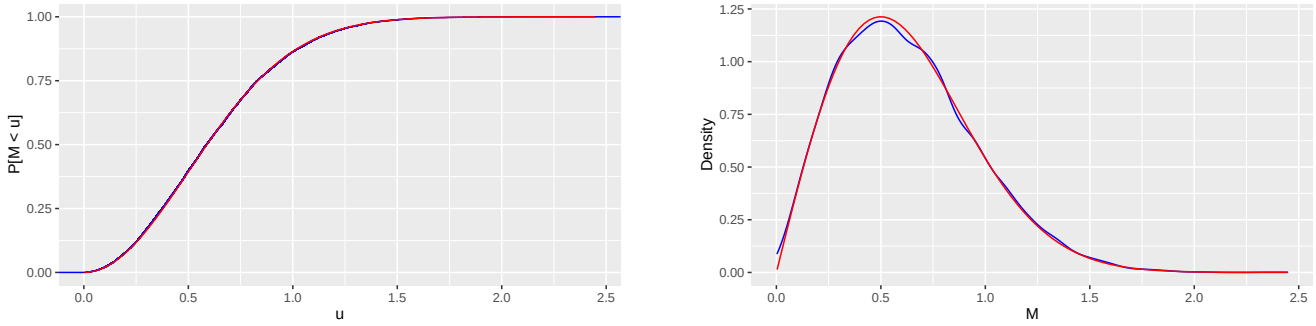


Figure 3.2: Empirical distribution of M making use of simulations of Brownian Bridges

Figure 3.2 shows that M is positive which makes sense since $M = \max X_t \geq X_0 = B_0 = 0$. One could also verify that M is distributed according to Rayleigh distribution, *i.e.*

$$\mathbb{P}[M \leq u] = 1 - e^{-2u^2}$$



(a) Cumulative distribution function

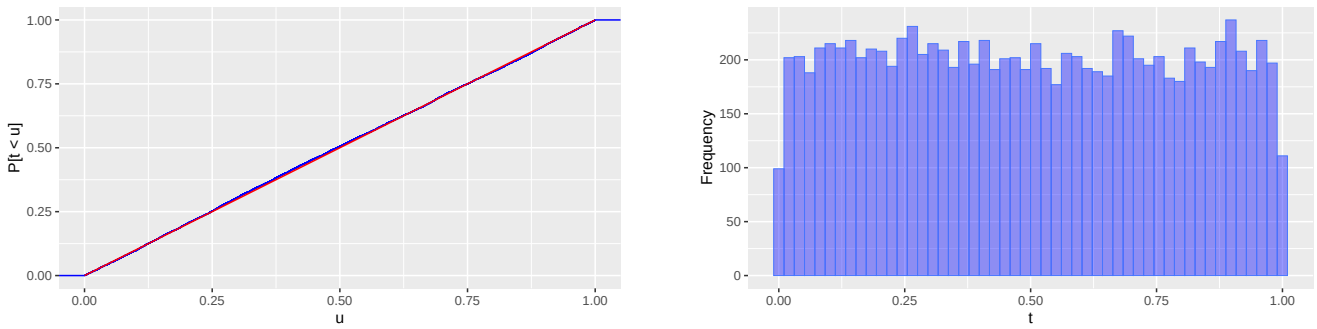
(b) Probability density function

Figure 3.3: Comparison between M empirical distribution (in blue) and the Rayleigh distribution (in red).

As Figure 3.3 shows, M seems distributed according to the Rayleigh distribution since the empirical cumulative distribution function of M samples in blue is very close to the theoretical in red, as well as for the probability density function (in fact the kernel density here). To assess this quantitatively, one could do a Kolmogorov-Smirnov test with null hypothesis H_0: M is distributed according to the Rayleigh distribution, H_1: M is not distributed according to the Rayleigh distribution. Such a test in `R` on the generated samples give a D statistic of 0.0071 and an associated p-value of 0.6993. Therefore, using a 0.05 threshold for the p-value, one may fail to reject H_0, that is M is distributed according to the Rayleigh distribution.

(b) Distribution of the time at which X_t attains its maximum

One may focus not only on the distribution of the maximum but also on the time at which X_t attains its maximum $t_M = \arg\max_{t\in[0,1]} X_t$. Therefore, the `R` function used to sample M is slightly modified in order to sample the time t_M at which X_t attains its maximum.



(a) Cumulative distribution function

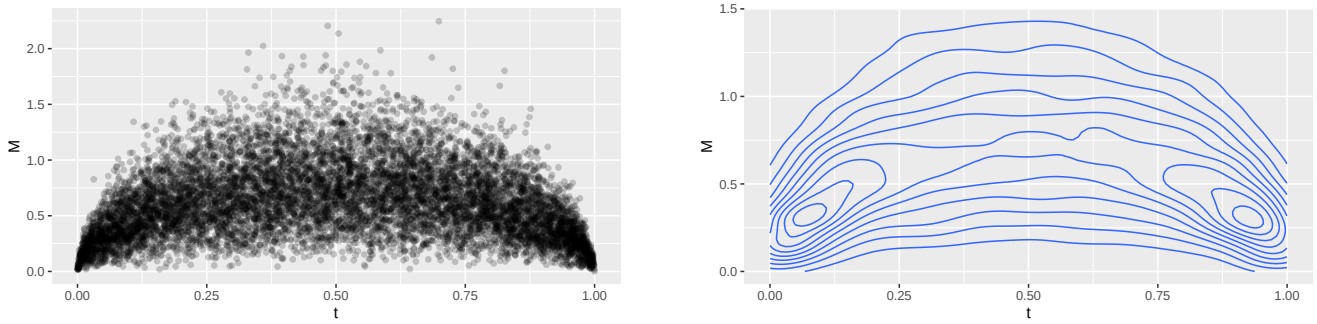
(b) Histogram

Figure 3.4: Comparison between t_M empirical distribution (in blue) and the uniform on $[0,1]$ distribution (in red).

As shown on Figure 3.4, the time at which X_t attains its maximum seems uniformly distributed on $[0,1]$, *i.e.* there is no more likely time $t \in [0,1]$ at which X_t is maximal. To assess this quantitatively, one could proceed to a Kolmogorov-Smirnov test with null hypothesis X_0: $t_M \sim \mathcal{U}([0,1])$, H_1: t_M is not distributed according to a uniform distribution on $[0,1]$. In `R`, the obtained D statistic is 0.0089 and the associated p-value is 0.4067: therefore, using a 0.05 threshold for the p-value, one may fail to reject the null hypothesis, *i.e.* t_M is distributed according to a uniform distribution on $[0,1]$.

(c) Joint distribution of the maximum and the time at which it is obtained

One can now focus on the joint distribution of the maximum and the time at which it is obtained, that is the distribution of (t_M, M). To do so, one could simulate $K = 10^4$ Brownian Bridges and plot their maxima according to the time at which the maxima are obtained.



(a) Empirical joint distribution

(b) Contour plot

Figure 3.5: Empirical joint distribution of time maxima and the time at which they are obtained from Brownian Bridges simulations.

As Figure 3.5 shows, this joint distribution is symmetrical to the axis $t_M = 0.5$ and has 2 modes. Moreover maxima attained near the middle ($t_M = 0.5$) are higher than maxima attained near the ends ($t_M = 0$ and $t_M = 1$) since the Brownian Bridges end points are constrained to be 0. In fact, it can be shown (Karatzas and Shreve, [2]) that the joint distribution density is

$$f_{M,t_M}(m,t) = \sqrt{\frac{2}{\pi}} \frac{m^2}{(t(1-t))^{\frac{3}{2}}} \exp\left(-\frac{m^2}{2t(1-t)}\right)$$

(d) Hypothesis testing

Let $\{X_i\}$ be a set of measurements and suppose we wish to test the claim that they are independent samples from a uniform distribution $\mathcal{U}([0,1])$ with distribution function $F = \mathbb{P}[X_i \leq t]$ such that

$$F(t) = \begin{cases} 0 & \text{if } t < 0 \\ t & \text{if } t \in [0,1] \\ 1 & \text{if } t > 1 \end{cases}$$

We can estimate the cumulative distribution function of the measurements using the empirical distribution function $F_n(t)$ defined as

$$F_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{[X_i \leq t]}$$

which is a random function of t. Kolmogorov and Smirnov derived a test for this hypothesis focusing on the probability distribution of the statistic

$$Y_n = \max_s \sqrt{n} |F_n(s) - F(s)|$$

that is, the maximum vertical distance between the empirical CDF and the true CDF under the hypothesis that $X_1, ..., X_n$ are independent samples from a distribution with CDF F.

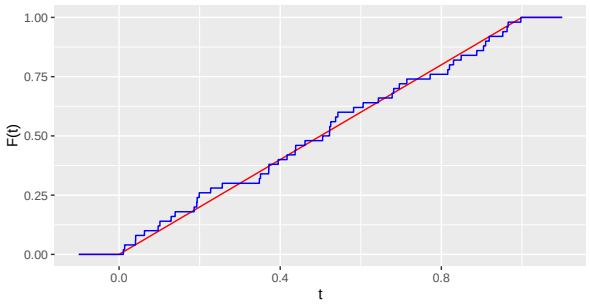


Figure 3.6: Comparison between the true CDF F (in red) and the empirical CDF F_n (in blue). The maximal vertical distance between the two CDF is attained at $t = 0.1990$.

It can therefore be shown (Lecture Notes [1]) that

$$\mathbb{E}[F_n(t)] = F(t)$$
$$\mathbb{E}[(F_n(s) - F(s))(F_n(t) - F(t))] = \frac{1}{n}(\min(F(s), F(t)) - F(s)F(t))$$

Hence, $\sqrt{n}|F_n(t) - F(t)|$ has the same mean and covariance function as $Q(F(t))$ for a Brownian Bridge $Q(\cdot)$. By Central Limit Theorem, the finite-dimensional distributions of $\sqrt{n}(F_n(t) - F(t))$ converges weakly to the normal distribution as $n \to \infty$.

Then, one could show using a functional central limit theorem (as Donkser's theorem) that Y_n converges to $Y = \max_{0 \leq t \leq 1} |Q(F(t))|$, the Kolmogorov-Smirnov statistic. This provides a non-parametric test which can be used to compare a sample with a reference probability distribution. Under the null hypothesis that $X_1, ..., X_n$ are sampled from a $\mathcal{U}([0,1])$ distribution with CDF F,

$$\sqrt{n} Y_n \xrightarrow{\mathcal{D}} \max_t |Q(F(t))|$$

For large n, the test statistic can be approximately estimated as

$$Y_n \simeq \frac{1}{\sqrt{n}} \max_t |Q(F(t))|$$

Therefore, from the provided dataset `q3.txt`, one can easily compute in `R` the test statistic Y_n using $Y_n = \max_s \sqrt{n}|F_n(s) - F(s)|$ since n, F_n and F are known: the calculated value is $Y_n = 0.0610$. Here in

our particular case, since $F(\cdot)$ is the identity function on $[0,1]$, there is no need to compute a Brownian Bridge with transformed time $F(t)$.

Then, one could create a `R` function which generates a standard Brownian Bridge X_t on $[0,1]$, applies an absolute value transformation to the Bridge and returns the maximum value attained by this transformed Brownian Bridge. The outputs of this function are therefore random since the Brownian Bridges are all different. Then, one could generate $K = 10^3$ realisations of this algorithm $\tilde{Y}_1, ...\tilde{Y}_K$ and compare it to the Y_n value computed previously. This would lead to the hypothesis testing p-value

$$p = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}_{[Y_n \leq \tilde{Y}_k]}$$

In `R`, the obtained p-value is 0.9870. Therefore, using a 0.05 threshold, one could fail to reject the null hypothesis, *i.e.* samples $X_1, ..., X_n$ from `q3.txt` are independent samples from a $\mathcal{U}([0,1])$ distribution. To verify this result, one could use the built-in `ks.test` function in `R`: the obtained D statistic is 0.0610 and the p-value is 0.9867, which are the same as obtained when doing the test manually.

The `R` code used for this question can be found below.

```
# Exercise 3

# (a)
generate.BM <- function(n=1e4, min=0, max=1){
  B <- c(0)
  delta_t <- (max - min)/n
  jumps <- rnorm(n-1, mean=0, sd=sqrt(delta_t))
  B <- c(B, cumsum(jumps))
  return(B)
}

n <- 1e4
t <- seq(0, 1, length.out=n)
B <- generate.BM()

ggplot() +
  geom_line(aes(x=t, y=B))

generate.BB <- function(n=1e4, min=0, max=1){
  B <- generate.BM(n, min, max)
  t <- seq(min, max, length.out=n)
  X <- B - t * B[length(B)]
  return(X)
}

X <- generate.BB()

ggplot() +
  geom_line(aes(x=t, y=X))

generate.M <- function(n=1e4, min=0, max=1, K=1e3){
  M_list <- c()
  for(k in 1:K){
    X <- generate.BB(n, min, max)
    M_list <- c(M_list, max(X))
    print(k)
  }
  return(M_list)
}
```

```
prayleigh <- function(u){
  return(1-exp(-2*u^2))
}

drayleigh <- function(u){
  return(4*u*exp(-2*u^2))
}

M_samples <- generate.M(K=1e4)

ggplot() +
  geom_histogram(aes(x=M_samples), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  labs(x="M", y="Frequency")

ggplot() +
  stat_ecdf(aes(x=M_samples), colour="blue") +
  geom_function(fun=prayleigh, colour="red") +
  xlab("u") + ylab("P[M < u]")

ggplot() +
  geom_density(aes(x=M_samples), colour="blue") +
  geom_function(fun=drayleigh, colour="red") +
  xlab("M") + ylab("Density")

ks.test(M_samples, prayleigh)

# (b)
generate.t <- function(n=1e4, min=0, max=1, K=1e3){
  t_list <- c()
  for(k in 1:K){
    X <- generate.BB(n, min, max)
    t <- (which.max(X) - 1) * (max - min)/n
    t_list <- c(t_list, t)
    print(k)
  }
  return(t_list)
}

t_samples <- generate.t(K=1e4)

ggplot() +
  geom_histogram(aes(x=t_samples), bins=50, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  xlab("t") + ylab("Frequency")

ggplot() +
  stat_ecdf(aes(x=t_samples), colour="blue") +
  geom_function(fun=punif, colour="red") +
  xlab("u") + ylab("P[t < u]")

ks.test(t_samples, "punif")

# (c)
generate.joint.t.M <- function(n=1e4, min=0, max=1, K=1e3){
  t_list <- c()
  M_list <- c()
  for(k in 1:K){
    X <- generate.BB(n, min, max)
    M_list <- c(M_list, max(X))
    t <- which.max(X) * (max - min)/n
```

```
    t_list <- c(t_list, t)
    print(k)
  }
  joint <- cbind(t_list, M_list)
  colnames(joint) <- c("t", "M")
  return(joint)
}

joint <- generate.joint.t.M(K=1e4)

ggplot() +
  geom_point(aes(x=joint[,1], y=joint[,2]), alpha=1/5) +
  xlab("t") + ylab("M")

ggplot() +
  geom_density_2d(aes(x=joint[,1], y=joint[,2])) +
  xlab("t") + ylab("M")

# (d)
data <- unname(unlist(read.table("q3.txt")))

hist(data, breaks=20)

Fn <- function(t){
  return(mean(data <= t))
}

tt <- seq(-0.1, 1.1, length.out=1e4)
Fnt <- sapply(tt, Fn)
Ft <- sapply(tt, punif)
ggplot() +
  geom_line(aes(x=tt, y=Ft), colour="red") +
  geom_line(aes(x=tt, y=Fnt), colour="blue") +
  xlab("t") + ylab("F(t)")

tt <- seq(0, 1, length.out=1e5)
Yn <- max(sqrt(length(data)) * abs(sapply(tt, Fn) - punif(tt)))
Yn/sqrt(length(data))

# This function works the same but is slower: there is no need to apply F(t) transformation
generate.BB.Q <- function(n=1e4, min=0, max=1){
  B <- generate.BM(n, min, max)
  t <- seq(min, max, length.out=n)
  Ft <- sapply(t, punif)
  X <- B - Ft * B[length(B)]
  return(max(abs(X)))
}

generate.BB.Q <- function(n=1e4, min=0, max=1){
  X <- generate.BB(n, min, max)
  return(max(abs(X)))
}

bridges_max <- c()
for(k in 1:1e3){
  bridges_max <- c(bridges_max, generate.BB.Q())
}

mean(Yn <= bridges_max)

ks.test(data, punif, exact=TRUE)
```

References

[1] Dr Andrew. Duncan (March 2022) *MATH70082 - Stochastic Processes 2021-2022 Lecture Notes*, Imperial College London MSc Statistics resources

[2] Ioannis Karatzas, Steven E. Shreve, *Brownian Motion and Stochastic Calculus*, 2012

[3] Joris Bierkens (University of Warwick), Andrew Duncan (Imperial College London), *Limit theorems for the Zig-Zag process*, 2017