Tutorial 4 – Giovanni Filomeno - 12315325

Exercise 18

a) Homogeneous Poisson Process

My code for generating a path of the homogeneous Poisson process is presented in Figure 1.

```r
## Part A: single Poisson-process path
simulate_poisson_path <- function(lambda, T_end) {
  t  <- 0                        # current time
  N  <- 0                        # current count
  ts <- c(0)                     # times at which N jumps
  Ns <- c(0)                     # values of N just after each jump

  ## keep generating exponential waiting times until we pass T_end
  while (TRUE) {
    w <- rexp(1, rate = lambda) # inter-arrival ~ Exp(λ)
    t <- t + w
    if (t > T_end) break
    N <- N + 1
    ts <- c(ts, t)
    Ns <- c(Ns, N)
  }

  ## pad the end so we can draw a flat line out to T_end
  ts <- c(ts, T_end)
  Ns <- c(Ns, N)

  data.frame(time = ts, count = Ns)
}
```

Figure 1: Homogeneous Poisson process

b) Plot a path for $\lambda = 3$ and $T = 10$

Figure 2 represents my result for the Poisson path.

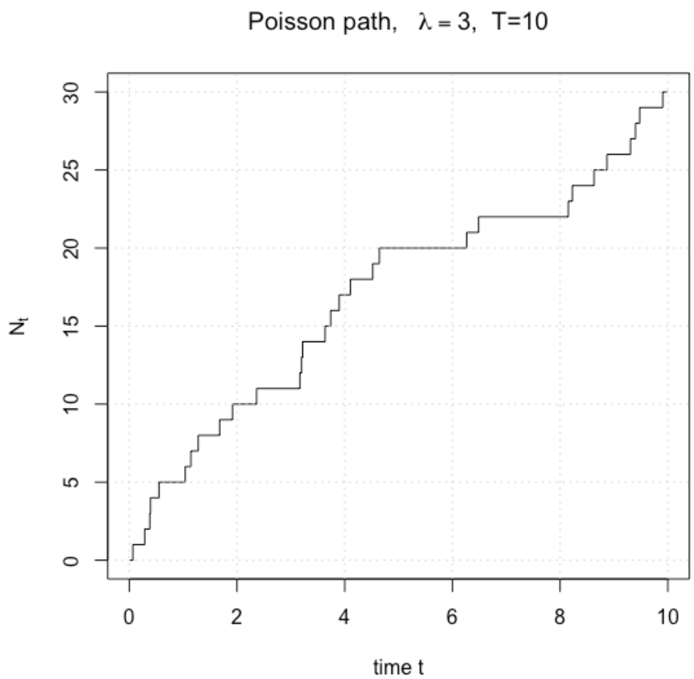

Figure 2: Homogeneous Poisson path for $\lambda = 30$ and $T = 10$

c) Empirical pmf vs simulated of random variable N_2

My algorithm for comparing the two pmf is presented in Figure 3, while the obtained result is in Figure 4. The empirical probability mass function (pmf) of N_2, obtained via Monte Carlo simulation, closely matches the theoretical Poisson pmf with parameter $\lambda \cdot 2 = 6$. This agreement confirms the validity of the simulation approach in approximating the distribution of the Poisson process at time $t = 2$.

```r
## Part B: plot the path for λ = 3, T = 10
set.seed(42)
path <- simulate_poisson_path(lambda, T_tot)

plot(path$time, path$count,
     type = "s",
     xlab = "time t",
     ylab = expression(N[t]),
     main = bquote("Poisson path,  "~lambda==.(lambda)*",  T="*.(T_tot)))
grid()

## Part C: N_2 empirical pmf vs. true pmf
sim_N2 <- rpois(M, lambda * 2)

## empirical pmf
pmf_hat <- table(sim_N2) / M

## plot empirical pmf
barplot(pmf_hat,
        beside = TRUE,
        names.arg = names(pmf_hat),
        xlab = expression(k),
        ylab = "Probability",
        main = expression("Empirical pmf of  N"[2]))

## overlay the true pmf
k_vals <- as.integer(names(pmf_hat))
points(1:length(k_vals),
       dpois(k_vals, lambda * 2),
       pch = 16, col = "red")
legend("topright", legend = c("Empirical", "True Poisson"),
       pch = c(22, 16),
       pt.bg = c("grey", NA),
       col = c("black", "red"),
       bty = "n")
```

Figure 3: Empirical pmf vs simulated algorithm – ex18c

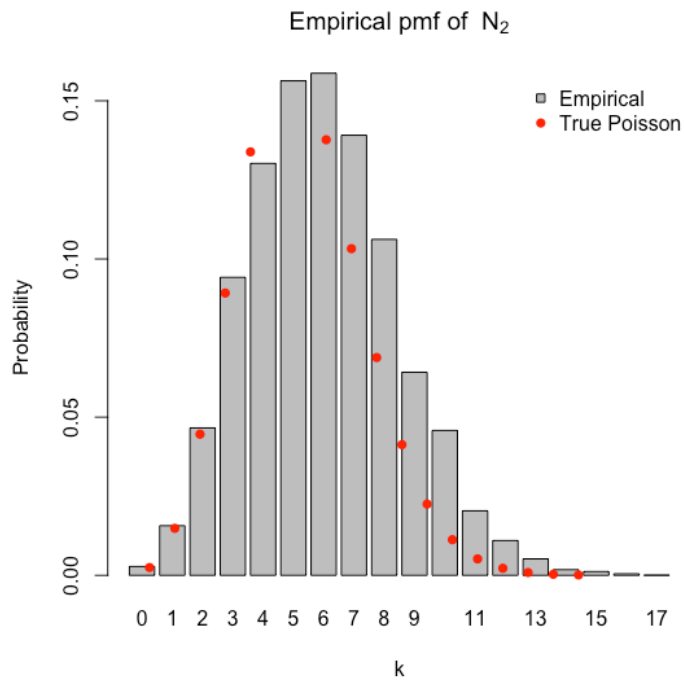

Figure 4: Empirical pmf vs simulated results – ex18c

d) Empirical pmf vs simulated of jumping time T_4

My algorithm for comparing the two pmf is presented in Figure 5, while the obtained result is in Figure 6. Also in this case, the empirical distribution of the fourth jump time T_4 aligns closely with the theoretical Gamma distribution, confirming the known result that the n-th jump time in a Poisson process follows a Gamma distribution with shape n and rate λ.

```
## Part D: T_4 empirical pdf vs. true pdf
sim_T4 <- rgamma(M, shape = 4, rate = lambda)

## empirical density estimate
hist(sim_T4,
     breaks = 60, probability = TRUE,
     xlab = expression(t),
     main = expression("Jump time  T"[4]*": empirical pdf vs. true pdf"))
# overlay true Gamma density
curve(dgamma(x, shape = 4, rate = lambda),
      from = 0, to = max(sim_T4),
      add = TRUE, lwd = 2, n = 1000, col = "red")
legend("topright", legend = c("Histogram", "True Γ(4,λ)"),
       fill = c("grey", NA), border = c("black", NA), lwd = c(NA, 2),
       col = c("black", "red"), bty = "n")
```

Figure 5: Empirical pmf vs simulated algorithm – ex18d

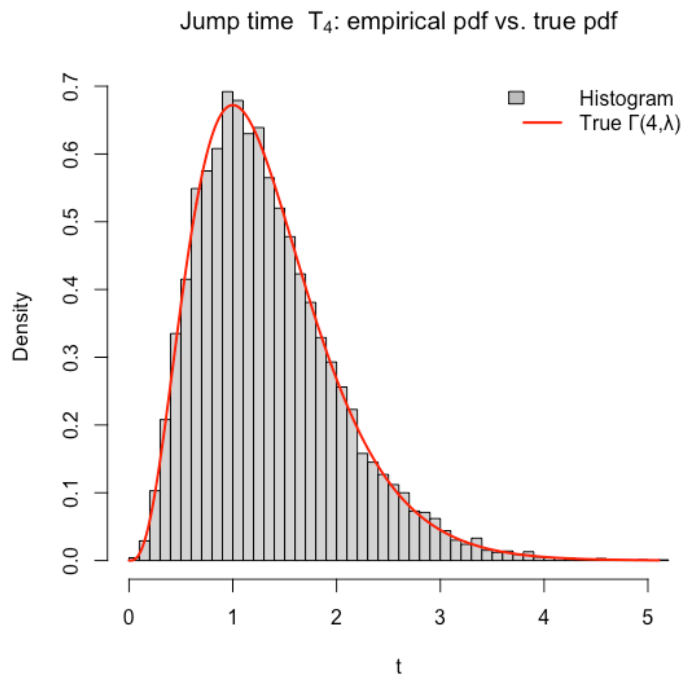

Figure 6: Empirical pmf vs simulated results – ex18d

Exercise 19

a) Algorithm for non-homogeneous Poisson process

My code for generating a path of the non-homogeneous Poisson process is presented in Figure 7.

```r
## Part A: generating path
simulate_NHPP_path <- function(T_end,
                               lambda_fun  = lambda_t,
                               lambda_max  = 5)
{
  t  <- 0           # current time
  N  <- 0           # current count
  ts <- c(0)        # jump times (incl. origin)
  Ns <- c(0)        # counts     (incl. N₀ = 0)

  while (TRUE) {
    w <- rexp(1, rate = lambda_max)
    t <- t + w
    if (t > T_end) break

    ## thinning step
    if (runif(1) < lambda_fun(t) / lambda_max) {
      N  <- N + 1
      ts <- c(ts, t)
      Ns <- c(Ns, N)
    }
  }

  ts <- c(ts, T_end)
  Ns <- c(Ns, N)

  data.frame(time = ts, count = Ns)
}
```

Figure 7: Non-homogeneous Poisson process

b) Plot a path for $T = 30$

The requested path is presented in Figure 8.

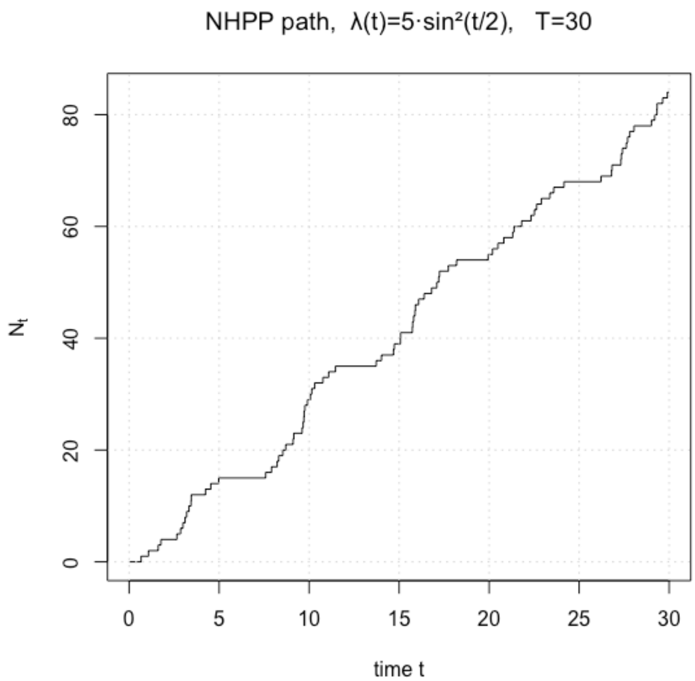

Figure 8: Non-homogeneous Poisson path for $T = 30$

c) Empirical vs Simulated pmf

In this case, the empirical vs simulated pmf presents a shift. This is an expected results when $\lambda(t)$ oscillates. Figure X shows the comparison.

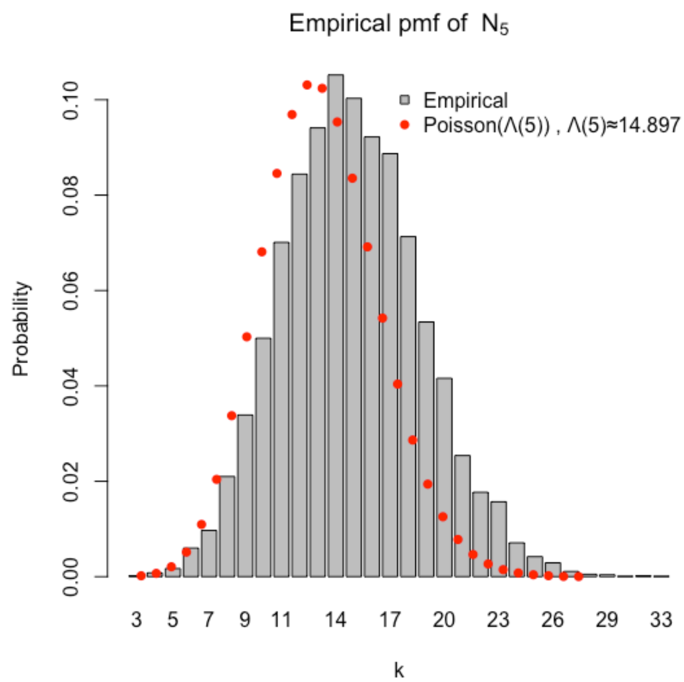

Figure 9: Empirical pmf vs simulated results – ex19c

Exercise 20

a) Path of Wiener process

My code for simulate the Wiener process is presented in Figure 10.

```
## Part A
simulate_wiener_drift <- function(mu      = 0,
                                  sigma   = 1,
                                  T_end   = 1,
                                  h       = 1e-3)
{
  n  <- ceiling(T_end / h)           # number of steps
  t  <- seq(0, T_end, length.out = n + 1)

  ## Brownian increments
  dW <- sqrt(h) * rnorm(n)
  W  <- c(0, cumsum(dW))

  X  <- mu * t + sigma * W

  data.frame(t = t, X = X, W = W)
}
```

Figure 10: Wiener process algorithm

b) 5 Wiener process

Figure 11 shows five independent sample paths for $\mu = 5, \sigma = 1$ over $[0,1]$ with $h = 10^{-3}$, and the red straight line is the theoretical mean $E[X_t] = 5t$.

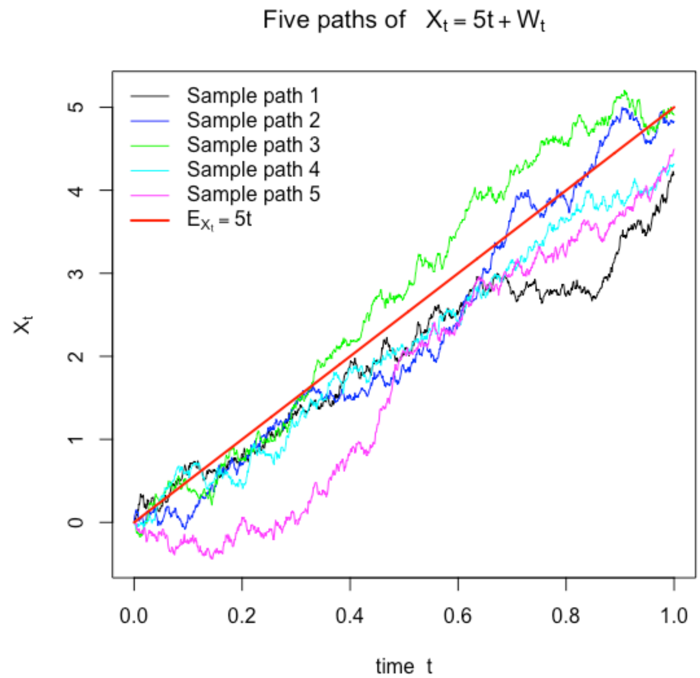

Figure 11: Five paths Wiener process

c) 5 Wiener process with different drift

Figure 12 presents one long path for each drift value $\mu \in \{0, -1, 1, -5, 5\}$ (all with $\sigma = 1$) over $[0,10]$. All curves share the same Brownian variability but diverge according to their deterministic drift.

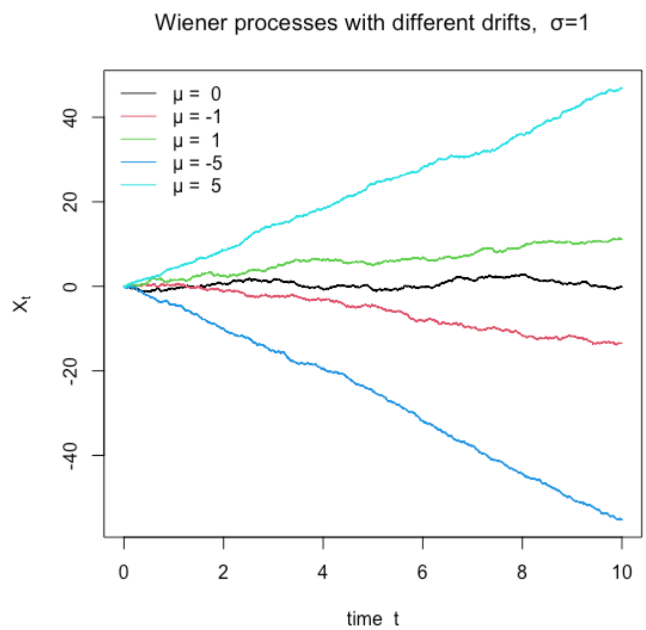

Figure 12: Five paths Wiener process with different drifts

d) 5 Wiener with different σ

The last figure (Figure 13) shows the requested simulation varying sigma.

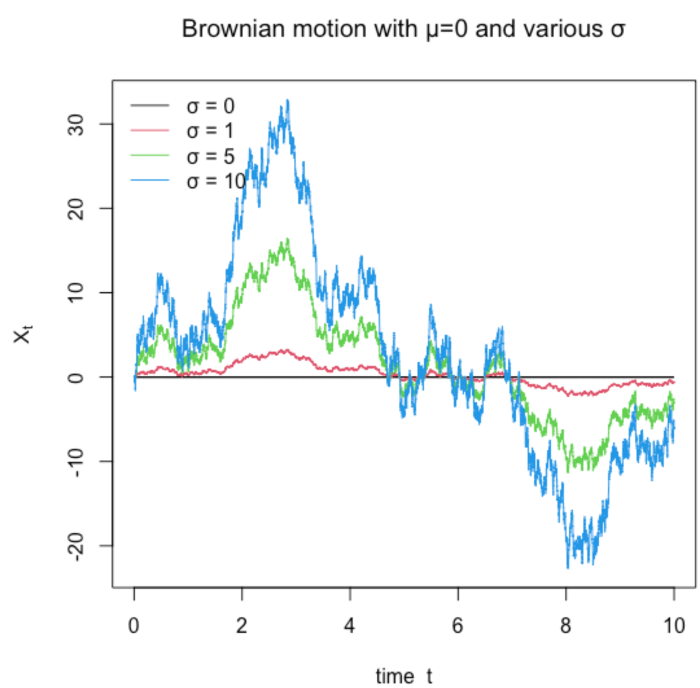

Figure 13: Five paths Wiener process with different sigma

Exercise 21

a) Geometric Brownian motion

My code for simulating the path of the geometric Brownian motion is presented in Figure 14.

```
## Part A: geometric brownian motion
simulate_gbm_exact <- function(mu      = 0,
                               sigma   = 1,
                               X0      = 1,
                               T_end   = 1,
                               h       = 1e-3)
{
  n  <- ceiling(T_end / h)             # steps
  t  <- seq(0, T_end, length.out = n + 1)

  ## Brownian increments
  dW <- sqrt(h) * rnorm(n)
  W  <- c(0, cumsum(dW))

  ## Exact GBM path
  X  <- X0 * exp((mu - 0.5 * sigma^2) * t + sigma * W)

  data.frame(t = t, X = X)
}
```

Figure 14: Geometric Brownian motion

b) 10 different Brownian paths

Figure 15 shows 10 independent geometric Brownian motion paths starting at $X_0 = 5$.

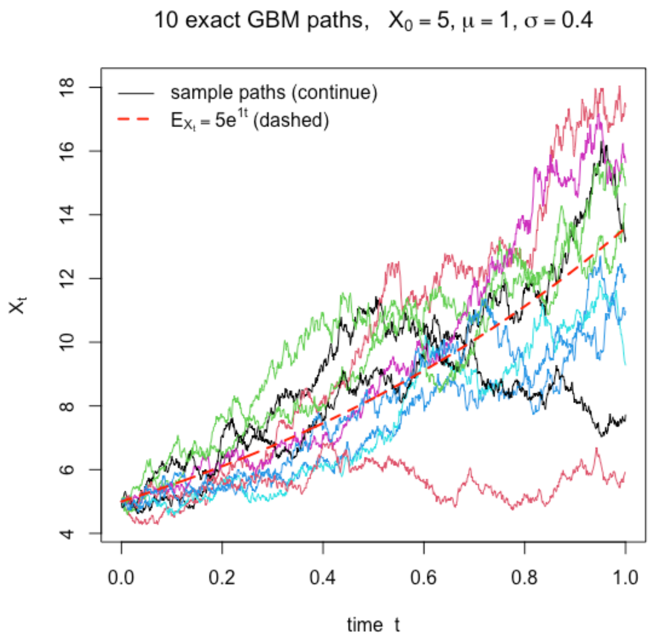

Figure 15: Different GBM paths

Exercise 22

a) Euler-Maruyama method

Figure 16 presents the code and the resulting plot for the Euler-Maruyama method.

```r
## Euler-Maruyama
X_EM <- numeric(n + 1)
X_EM[1] <- X0
for (k in 1:n) {
  X_EM[k + 1] <- X_EM[k] +
    mu    * X_EM[k] * h +
    sigma * X_EM[k] * dW[k]
}
```

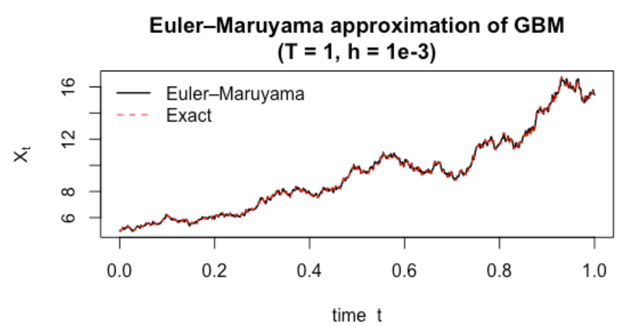

Figure 16: Euler-Maruyama code and plot

b) Milstein method

Figure 17 presents the code and the resulting plot for the Milstein method.

```
## Milstein
X_Mil <- numeric(n + 1)
X_Mil[1] <- X0
for (k in 1:n) {
  X_Mil[k + 1] <- X_Mil[k] +
    mu    * X_Mil[k] * h +
    sigma * X_Mil[k] * dW[k] +
    0.5 * sigma^2 * X_Mil[k]
    * (dW[k]^2 - h)   # Milstein
}
```

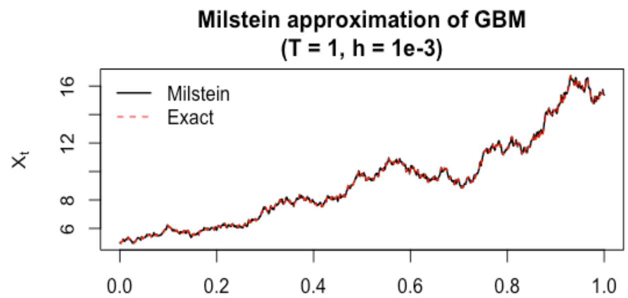

Figure 17: Milstein code and plot

Exercise 23

a) Reproducing the Week10 graph

Figure 18 represents a mimicking graph of Week10. A small shift in y is given to perfectly match the reference graph.

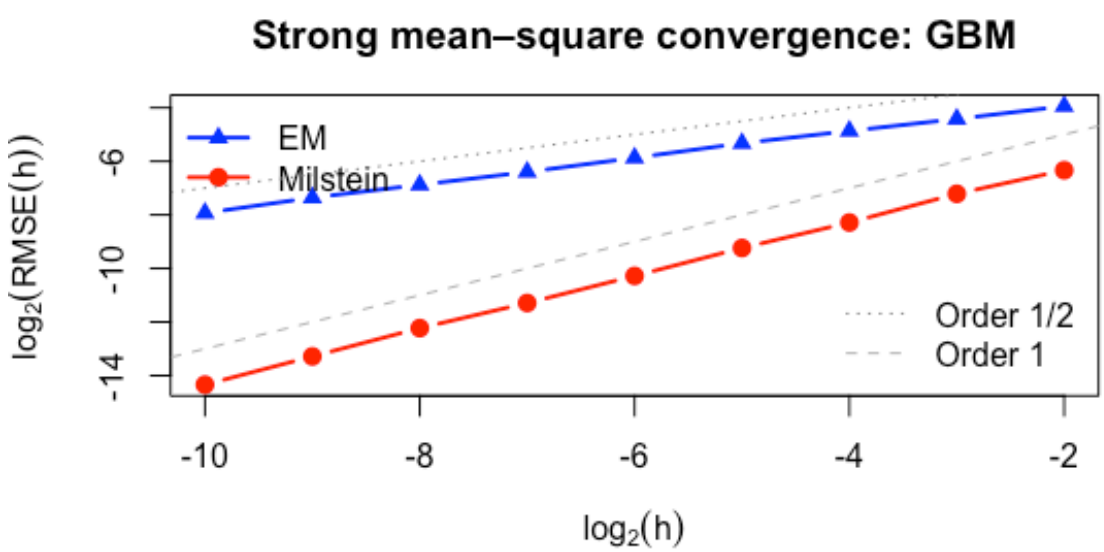

Figure 18: Week10 Graph mimic

Exercise 24

I decided to combine the entire exercise in one. Using the numerical integration, I obtained $\theta \approx 1.9600883115$ while using the Monte Carlo with $n = 10^4$ I obtained $\hat{\theta} \approx$

1.9560947793. Figure 19 shows how the estimation of $\hat{\theta}$ changes along the realization n. The red line represents the value of θ obtained via standard integration.

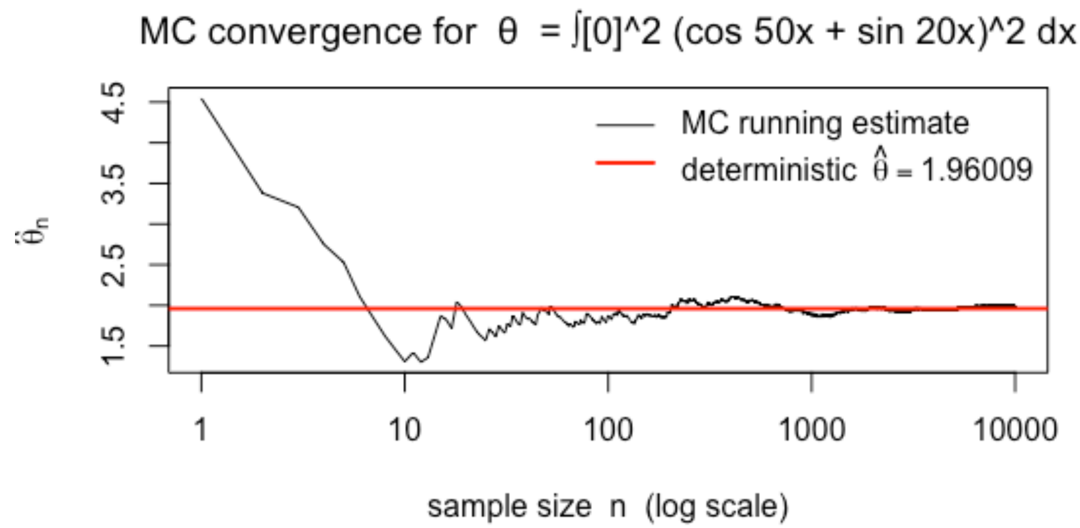

Figure 19: Standard integration vs Monte Carlo