Self Study_1

Exercise 1.

1) Itô's formel på $f(x,y) = \frac{x}{y}$

$$\Rightarrow \quad f(x,y) = \frac{x_0}{y_0} + \int_0^t \frac{1}{y}\, dx - \int_0^t \frac{x}{y^2}\, dy$$
$$+ \frac{1}{2}\int_0^t 0 \; d\langle X,X\rangle - \frac{1}{2}\int_0^t \frac{1}{y^2}\, d\langle X,Y\rangle$$
$$- \frac{1}{2}\int_0^t \frac{1}{y^2}\, d\langle Y,X\rangle + \frac{1}{2}\int_0^t \frac{2x}{y^3}\, d\langle Y,Y\rangle$$
$$= \frac{x_0}{y_0} + \int_0^t \frac{1}{y}\, dx - \int_0^t \frac{x}{y^2}\, dy - \int \frac{1}{y^2}\, d\langle X,Y\rangle$$
$$+ \int_0^t \frac{x}{y^3}\, d\langle Y,Y\rangle$$

Now let

$X_t := \frac{Z_t}{1+\int_0^t Z_s\, ds}$. Hence by Itô's lemma from before, and letting

$Y_t := 1 + \int_0^t Z_s\, ds$, we get:

$$X_t = \frac{Z_0}{1} + \int_0^t \frac{1}{Y_s}\, dZ_s - \int_0^t \frac{Z_s}{Y_s^2}\, dY_s$$
$$- \int \frac{1}{Y_s^2}\, d\langle Z,Y\rangle_s + \int_0^t \frac{Z_s}{Y_s^3}\, d\langle Y,Y\rangle_s$$

Now, notice that

$$\langle Y \rangle_t = \int_0^t z_s^2 \, d\langle S \rangle = 0.$$

$\Rightarrow \langle Z, Y \rangle = 0.$ Thus:

$$X_t = Z_0 + \int_0^t \frac{1}{Y_s} dZ_s - \int_0^t \frac{Z_s}{Y_s^2} dY_s$$

Recall $z_0 = x$, and $dZ_s = z_s \lambda ds + Z_s \bar{\sigma} dB_s$ gives

$$X_t = x + \int_0^t \lambda \frac{Z_s}{Y_s} ds + \int_0^t \bar{\sigma} \frac{Z_s}{Y_s} dB_s - \int_0^t \frac{Z_s}{Y_s^2} dY_s$$

Recall definition of Y_s, and $\frac{Z_s}{Y_s} = X_s$:

$$X_t = x + \int_0^t \lambda X_s ds + \int_0^t \bar{\sigma} X_s dB_s - \int_0^t \frac{Z_s}{Y_s^2} d\left(\int_0^s z_r dr\right)$$

$$= x + \int_0^t \lambda X_s ds + \int_0^t \bar{\sigma} X_s dB_s - \int_0^t X_s^2 ds$$

$$= x + \int_0^t (\lambda X_s - X_s^2) ds + \int_0^t \bar{\sigma} X_s dB_s$$

This implies:

$$dX_t = (\lambda X_s - X_s^2) ds + \bar{\sigma} X_s dB_s$$

Strong solution argument:

It is a strong solution, since X is determined with respect to the same Brownian motion B as the one defining the SDE, and thus X is adapted to the natural filtration of B.

Exercise 2.

Parameters:

$$\lambda = x = \bar{\sigma} = T = 1.$$

1, 2

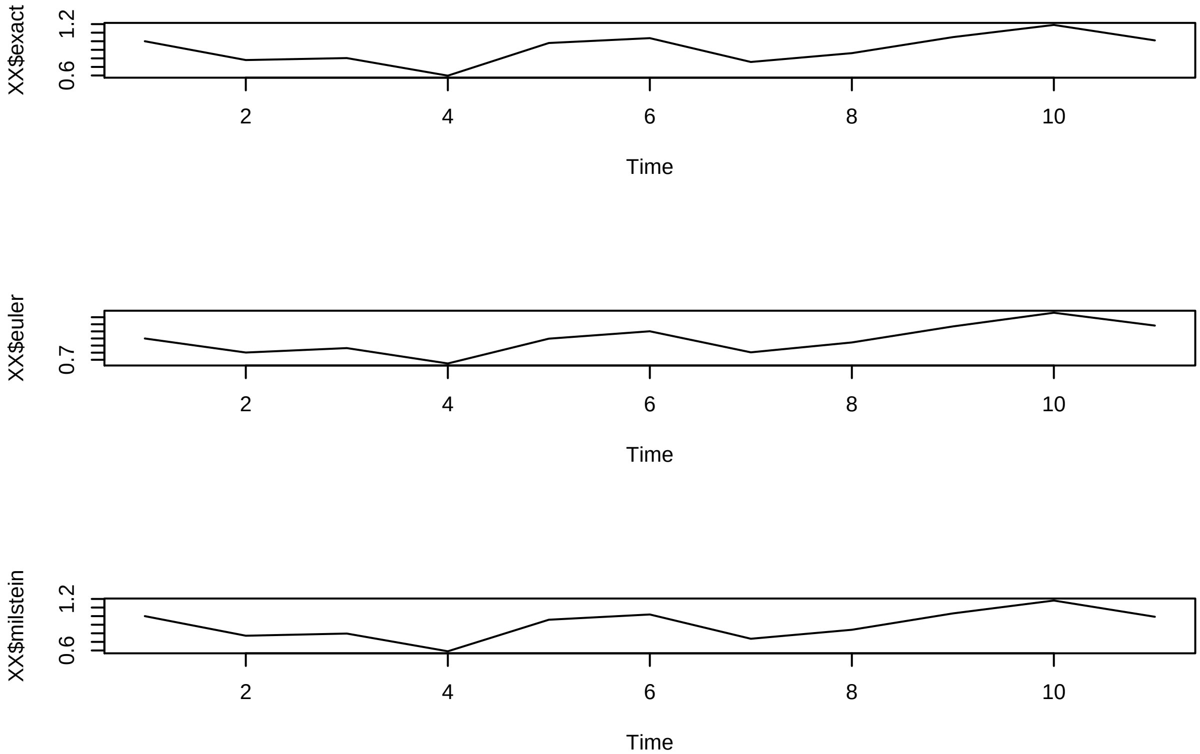
By Using a Riemann approximation of $\int_0^t Z_s ds$, write a code that simulates (with error) the exact solution of (1) (i.e. (2)) on an equally spaced grid of $[0,1]$ given by

$$t_k = \frac{k}{n}, \quad k = 0, \dots, n, \quad n \in \mathbb{N}.$$

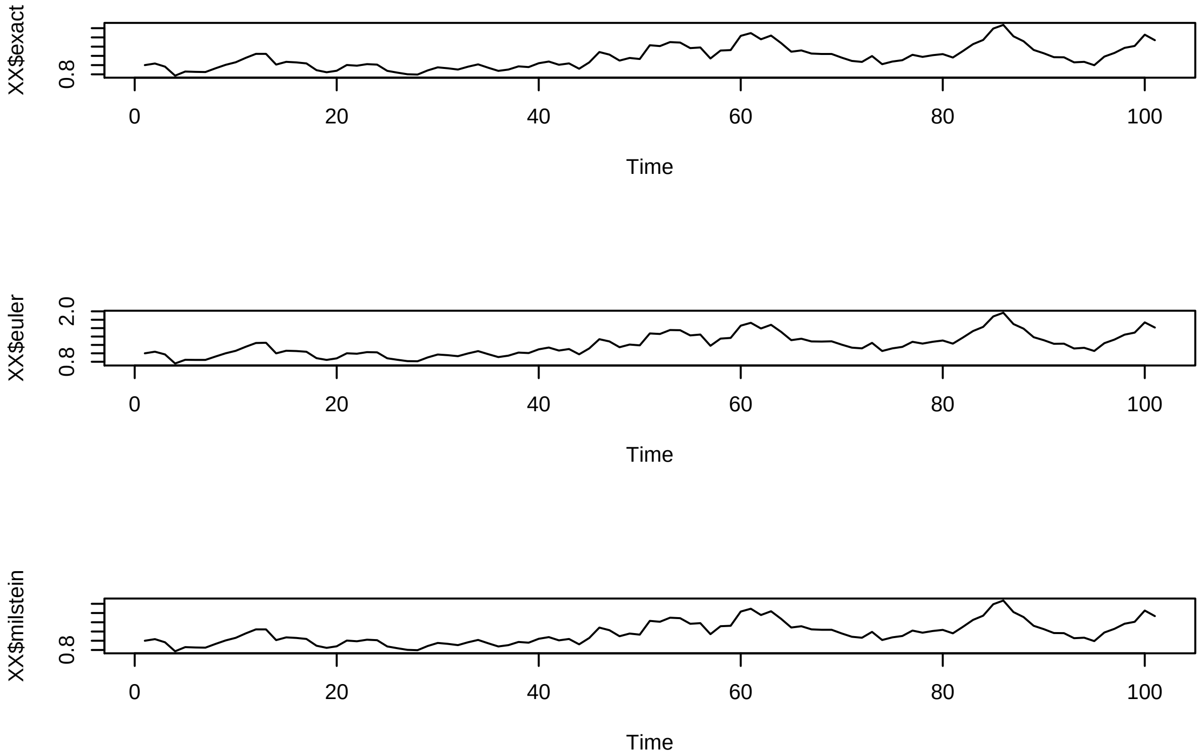
Plot simulated paths for $n = 10000$.

```
set.seed(1)


sim_BM <- function(T_, delta) {
  tmesh <- seq(from = 0, to = T_, by = delta)
  N <- length(tmesh)
  step <- delta * T_
  Norm <- rnorm(N)
  B <- c(0, cumsum(sqrt(step) * Norm[-N]))
  output <- data.frame(B = B, Norm = Norm)
  return(output)
}

Z_t <- function(t, B) {
  exp(0.5 * t + B)
}

b <- function(t, X_t) {
  X_t - X_t^2
}

sigma <- function(t, X_t) {
  X_t
}

sigma_dif <- function(t, X_t) {
  1
}

approx_integral_exact_solution_Euler_sol_Milstein_sol <-
  function(delta, T_ = 1, b, sigma, x_0 = 0, Z_t, sigma_dif) {
```

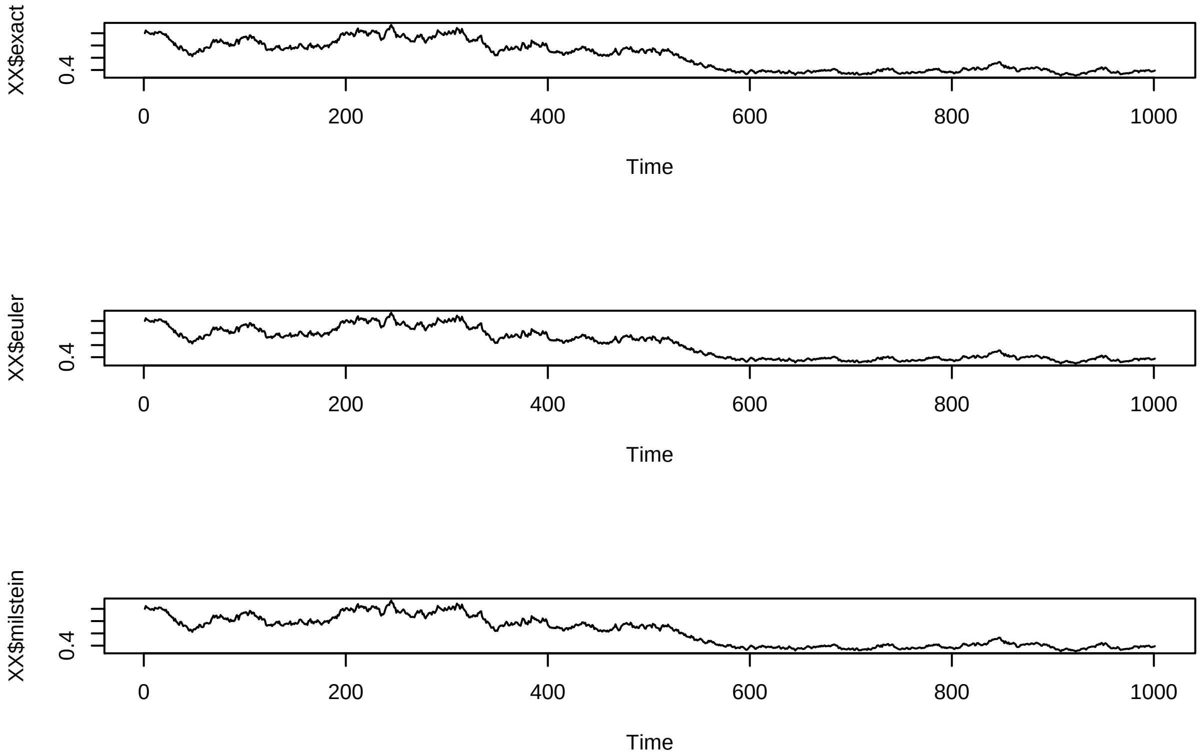
```r
  tmesh <- seq(from = 0, to = T_, by = delta)
  N_n <- length(tmesh)
  # N_n <- ceiling(T_ / delta)
  sim_BM_result <- sim_BM(T_, delta)
  B <- sim_BM_result$B
  X <- c(0)
  X_2 <- c(x_0)
  X_3 <- c(x_0)
  step <- delta * T_
  for (k in 2:(N_n)) {
    # Approximate integral
    X[k] <- X[k - 1] + Z_t(tmesh[k], B[k - 1]) * step
    # Euler scheme
    X_2[k] <- X_2[k - 1] + b(tmesh[k], X_2[k - 1]) * step +
      sigma(tmesh[k], X_2[k - 1]) * (B[k] - B[k - 1])
    # Milstein scheme
    X_3[k] <- X_3[k - 1] + b(tmesh[k], X_3[k - 1]) * step + sigma(tmesh[k], X_3[k - 1]) *
      (B[k] - B[k - 1]) +
      1 / 2 * sigma(tmesh[k - 1], X_3[k - 1]) *
      sigma_dif(tmesh[k - 1], X_3[k - 1]) * ((B[k] - B[k - 1])^2 - step)
  }

  XX <- Z_t(t = tmesh, B = B) / (1 + X)
  output <- data.frame(
    exact = XX,
    euler = X_2,
    milstein = X_3
  )
  return(output)
}


plot_sde <- function(n) {
  XX <- approx_integral_exact_solution_Euler_sol_Milstein_sol(
    delta = 1 / n,
    T_ = 1,
    b = b,
    sigma = sigma,
    x_0 = 1,
    Z_t = Z_t,
    sigma_dif = sigma_dif
  )
  par(mfrow = c(3, 1))
  plot.ts(XX$exact)
  plot.ts(XX$euler)
  plot.ts(XX$milstein)
  par(mfrow = c(1, 1))
}


plot_sde(10)
```


XX$exact
1.2
0.6
2
4
6
8
10
Time
XX$euler
0.7
2
4
6
8
10
Time
XX$milstein
1.2
0.6
2
4
6
8
10
Time

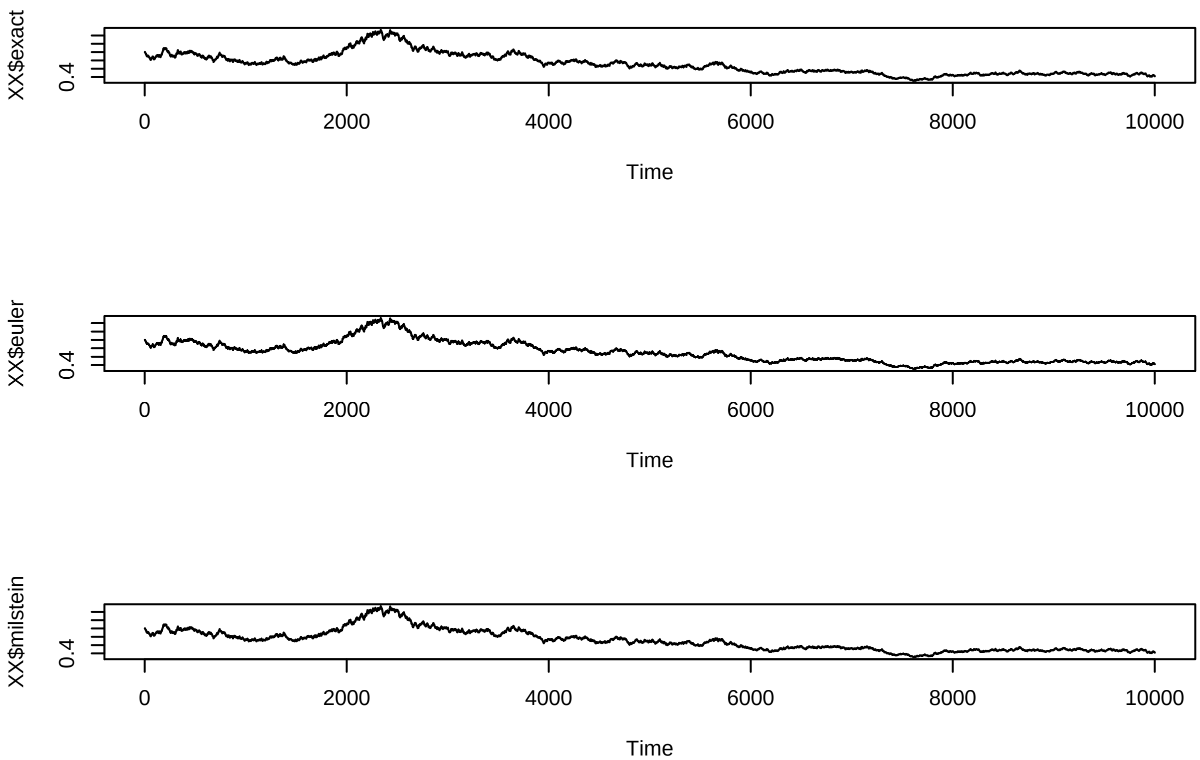
```
plot_sde(100)
```




```
plot_sde(1000)
```




```
plot_sde(10000)
```




```
# Exercise 6.
afstand_max <- function(a, b) {
  dist <- abs(a - b)^2
  max(dist)
}


MC_estimate_comparison <- function(n, MC_n) {
  MC <- data.frame()

  for (i in 1:MC_n) {
    output <- approx_integral_exact_solution_Euler_sol_Milstein_sol(
      delta = 1 / n,
      T_ = 1,
      b = b,
      sigma = sigma,
      x_0 = 1,
      Z_t = Z_t,
      sigma_dif = sigma_dif
    )
    MC[i, "Euler"] <- afstand_max(output$exact, output$euler)
    MC[i, "Milstein"] <- afstand_max(output$exact, output$milstein)
  }

  output_data <- data.frame(n = n,
                            Euler_mean_dist = mean(MC$Euler),
```

```
                                Milstein_mean_dist = mean(MC$Milstein))
  return(output_data)
}

comp_1 <- MC_estimate_comparison(1, 1000)
comp_1
```

```
##   n Euler_mean_dist Milstein_mean_dist
## 1 1       0.7693483          0.1919788
```

```
comp_10 <- MC_estimate_comparison(10, 1000)
comp_10
```

```
##    n Euler_mean_dist Milstein_mean_dist
## 1 10        0.080282         0.01188463
```

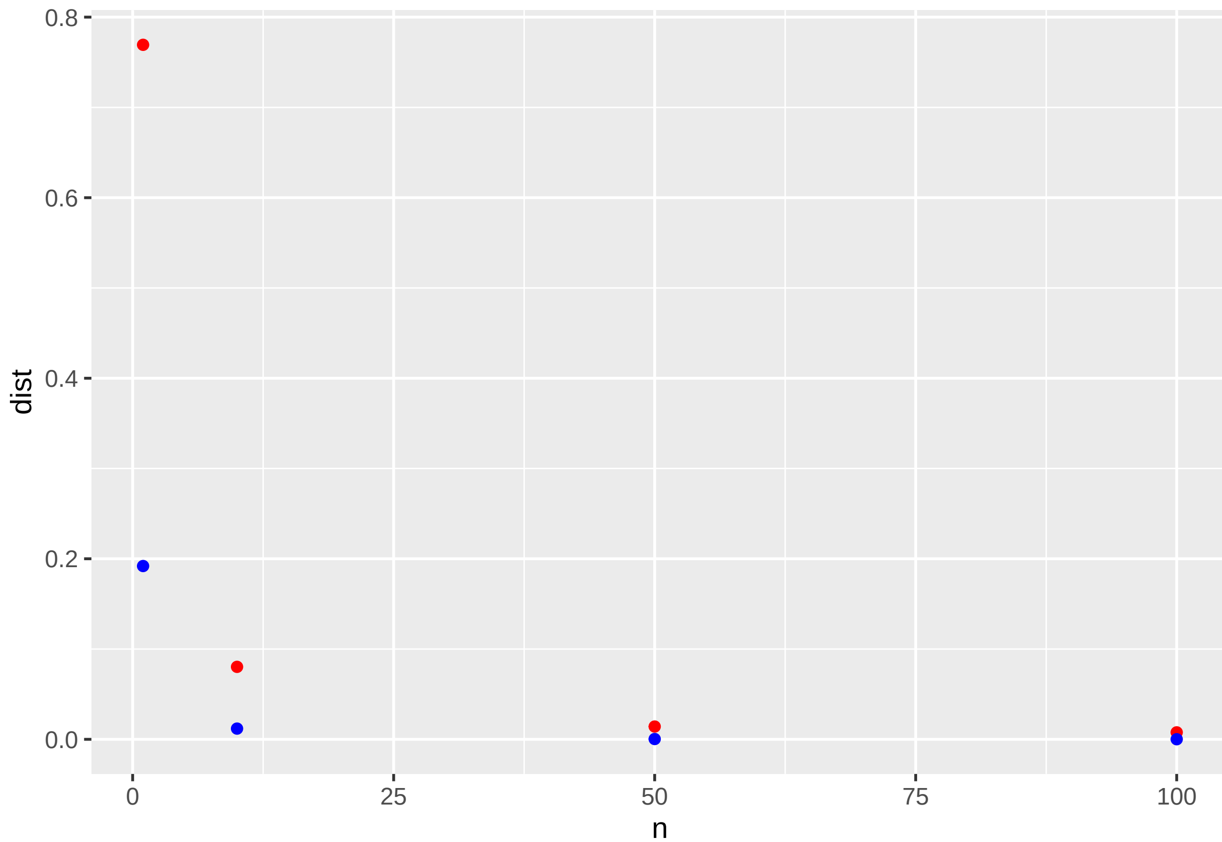
```
comp_50 <- MC_estimate_comparison(50, 1000)
comp_50
```

```
##    n Euler_mean_dist Milstein_mean_dist
## 1 50      0.01415563       0.0003219432
```

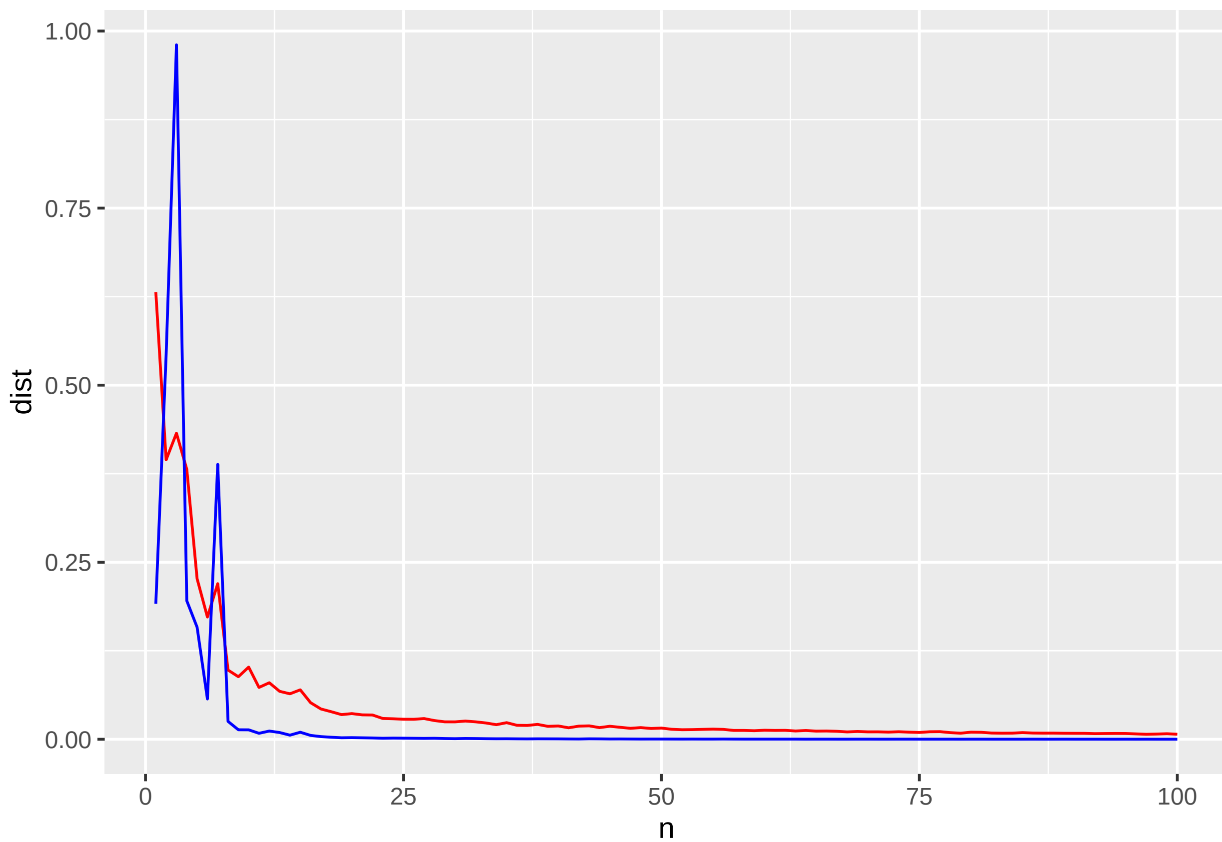
```
comp_100 <- MC_estimate_comparison(100, 1000)
comp_100
```

```
##     n Euler_mean_dist Milstein_mean_dist
## 1 100     0.007715346       0.0001038635
```

```
plot_data_mc_est_compa <- bind_rows(comp_1, comp_10, comp_50, comp_100)

ggplot(data = plot_data_mc_est_compa, aes(x = n)) +
  geom_point(aes(y = Euler_mean_dist), color = "red") +
  geom_point(aes(y = Milstein_mean_dist), color = "blue") +
  labs(y = "dist")
```




```
data <- data.frame()

for (i in 1:100) {
  MC_estimate_comparison_data <- suppressWarnings(MC_estimate_comparison(i,1000))
  data <- bind_rows(data, MC_estimate_comparison_data)
}

ggplot(data = data, aes(x = n)) +
  geom_line(aes(y = Euler_mean_dist), color = "red") +
  geom_line(aes(y = Milstein_mean_dist), color = "blue") +
  labs(y = "dist")
```

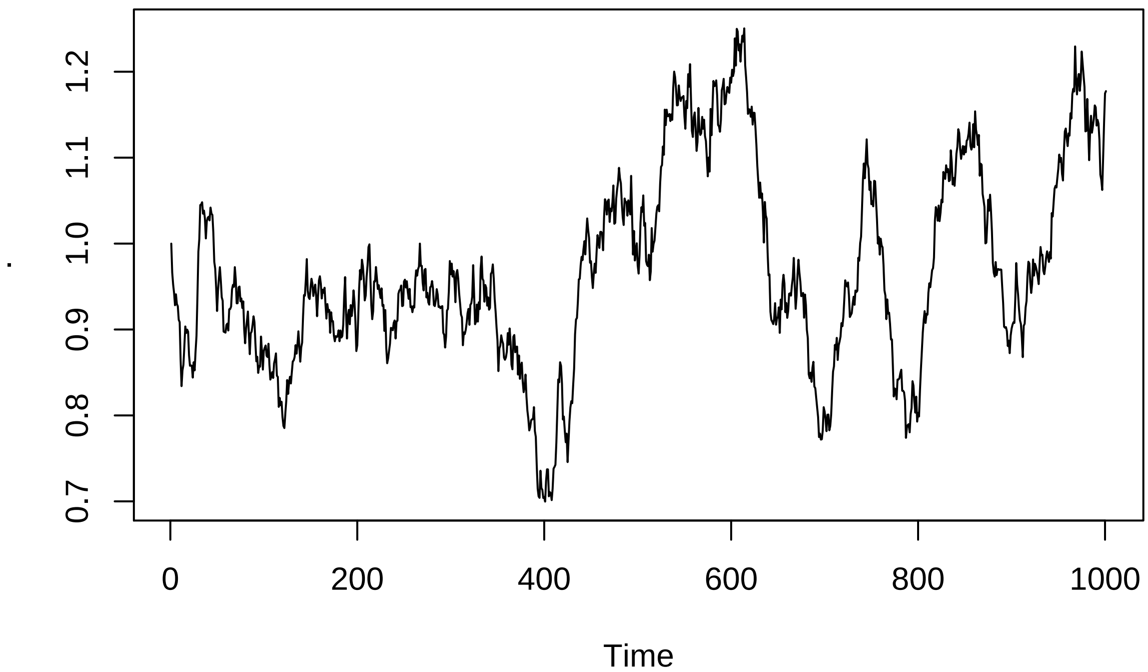


Exercise 3 and 4

Last part of exercise 4

```
b_V <- function(t, X_t) {
  0.5 - 0.5 * X_t
}

sigma_V <- function(t, X_t) {
  0.4 * sqrt(X_t)
}

b_S <- function(t, X_t) {
  0
}

sigma_S_1 <- function(t, X_t, V_t) {
  sqrt((1 - (-0.7)^2) * V_t) * X_t
}

sigma_S_2 <- function(t, X_t, V_t) {
  X_t * sqrt(V_t) * (-0.7)
}
```

```r
Euler_scheme <- function(delta, T_ = 1) {
  N_n <- ceiling(T_ / delta)
  tmesh <- seq(from = 0, to = T_, by = delta)
  sim_BM_result_1 <- sim_BM(T_, delta)
  sim_BM_result_2 <- sim_BM(T_, delta)
  B_1 <- sim_BM_result_1$B
  B_2 <- sim_BM_result_2$B
  X <- c(0.3)
  X_2 <- c(1)
  step <- delta * T_
  for (k in 2:(N_n + 1)) {
    # V
    X[k] <- X[k - 1] + b_V(tmesh[k], X[k - 1]) * step +
      sigma_V(tmesh[k], X[k - 1]) * (B_2[k] - B_2[k - 1])
  }

  for (k in 2:(N_n + 1)) {
    X_2[k] <- X_2[k - 1] + b_S(tmesh[k], X_2[k - 1]) * step +
      sigma_S_1(tmesh[k], X_2[k - 1], X[k - 1]) * (B_1[k] - B_1[k - 1]) +
      sigma_S_2(tmesh[k], X_2[k - 1], X[k - 1]) * (B_2[k] - B_2[k - 1])
  }

  return(X_2)
}

Euler_scheme(delta = 1 / 1000, T_ = 1) %>% plot.ts()
```




```
data_mean_Euler_scheme <- c()
for (i in 1:10000) {
  data_mean_Euler_scheme[i] <- max(last(Euler_scheme(delta = 1 / 1000, T_ = 1)) - 1, 0)
}
mean(data_mean_Euler_scheme)
```

```
## [1] 0.2551939
```