CEE290\_HW1

Gulnaz Shalgumbayeva

2024-04-14

***Problem 1***

# Define the matrix X  
X <- matrix(c(-2, 4, 1, 5, 0,  
 0, 2, 0, 4, 2,  
 -1, -2, 0, 3, 2,  
 2, 1, -4, -5, -1,  
 0, -2, -3, 0, 0), byrow = TRUE, nrow = 5)  
  
# Define the vector d̃  
d\_tilde <- c(-26, -18, -29, 54, 1)  
  
# Calculate the least squares solution theta\_hat  
# This is done by solving the normal equation (X^T X) theta = X^T d\_tilde  
theta\_hat <- solve(t(X) %\*% X) %\*% t(X) %\*% d\_tilde  
  
# Output theta\_hat  
theta\_hat

## [,1]  
## [1,] 2  
## [2,] 4  
## [3,] -3  
## [4,] -7  
## [5,] 1

Checking with lm function

# Create a data frame representing the system  
df <- data.frame(a = c(-2, 0, -1, 2, 0),  
 b = c(4, 2, -2, 1, -2),  
 c = c(1, 0, 0, -4, -3),  
 d = c(5, 4, 3, -5, 0),  
 e = c(0, 2, 2, -1, 0),  
 f = c(-26, -18, -29, 54, 1)) # f is the response  
  
# Use lm to solve it  
fit <- lm(f ~ a + b + c + d + e - 1, data = df) # '-1' to exclude the intercept  
  
# The coefficients should be close to the theta vector  
coef(fit)

## a b c d e   
## 2 4 -3 -7 1

# Define the matrix X  
X <- matrix(c(-2, 4, 1, 5, 0,  
 0, 2, 0, 4, 2,  
 -1, -2, 0, 3, 2,  
 2, 1, -4, -5, -1,  
 0, -2, -3, 0, 0), byrow = TRUE, nrow = 5)  
  
# Calculate the determinant of X  
determinant\_X <- det(X)  
  
# Output the determinant  
determinant\_X

## [1] 200

***Problem 2*** ***part 2.3***

# Define the vectors for time t and infiltration I  
t <- c(0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00)  
I <- c(0.051, 0.075, 0.123, 0.176, 0.264, 0.451, 0.682)  
  
# Define the design matrix X with sqrt(t) and t  
X <- cbind(sqrt(t), t)  
  
# Calculate the least squares solution using the normal equation  
theta\_hat\_normal <- solve(t(X) %\*% X) %\*% t(X) %\*% I  
  
# Print the solution with at least 3 digits for the fractional part  
print(theta\_hat\_normal, digits = 6)

## [,1]  
## 0.511503  
## t 0.171996

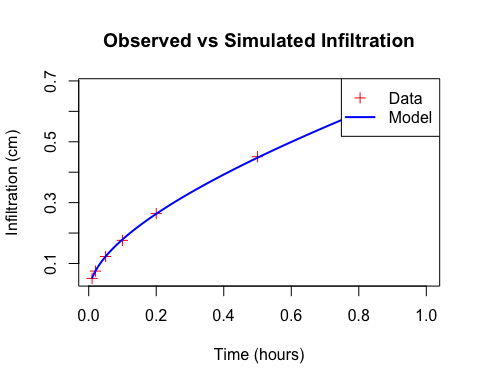
check with lm function

# We can use a data frame to store our variables for lm function  
data <- data.frame(t = c(0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00),  
 I = c(0.051, 0.075, 0.123, 0.176, 0.264, 0.451, 0.682),  
 sqrt\_t = sqrt(c(0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00)))  
  
# Perform the linear regression with lm  
# Notice that we use '0 +' to remove the intercept from the model  
# and 'sqrt\_t' to represent S\*sqrt(t) and 't' for Ks\*t  
fit <- lm(I ~ 0 + sqrt\_t + t, data = data)  
  
# Extract the coefficients for S (associated with sqrt\_t) and Ks (associated with t)  
theta\_hat\_lm <- coef(fit)  
  
# Print the solution with at least 3 digits for the fractional part  
print(theta\_hat\_lm, digits = 6)

## sqrt\_t t   
## 0.511503 0.171996

***part 2.4***

# Our estimated parameters  
S <-0.511503  
Ks <- 0.171996 # Ks  
  
# Original observed data  
t <- c(0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00)  
I\_observed <- c(0.051, 0.075, 0.123, 0.176, 0.264, 0.451, 0.682)  
  
# Create a sequence of time values for the simulation  
t\_sim <- seq(min(t), max(t), length.out = 100)  
  
# Simulate the infiltration curve using the computed coefficients  
I\_simulated <- S \* sqrt(t\_sim) + Ks \* t\_sim  
  
# Plot the observed data  
plot(t, I\_observed, xlab = "Time (hours)", ylab = "Infiltration (cm)",  
 pch = 3, col = "red", main = "Observed vs Simulated Infiltration")  
  
# Plot the simulated infiltration curve  
lines(t\_sim, I\_simulated, col = "blue", lwd = 2)  
  
# Add a legend  
legend("topright", legend = c("Data", "Model"),  
 col = c("red", "blue"), pch = c(3, NA), lty = c(NA, 1), lwd = c(NA, 2))



# Assuming you want to save the plot to a file  
png("infiltration\_plot.png")  
plot(t, I\_observed, xlab = "Time (hours)", ylab = "Infiltration (cm)",  
 pch = 3, col = "red", main = "Observed vs Simulated Infiltration")  
lines(t\_sim, I\_simulated, col = "blue", lwd = 2)  
legend("topright", legend = c("Data", "Model"),  
 col = c("red", "blue"), pch = c(3, NA), lty = c(NA, 1), lwd = c(NA, 2))  
dev.off()

## quartz\_off\_screen   
## 2

***part 2.5 and 2.6***

S <-0.511503  
Ks <- 0.171996 # Ks  
  
# Calculate the predicted values using the model  
I\_predicted <- S \* sqrt(t) + Ks \* t # my y hats   
  
# Calculate residuals  
residuals <- I\_observed - I\_predicted # y-yhat  
  
# Calculate the sum of squared residuals  
SSR <- sum(residuals^2)  
  
# The units for SSR will be the square of the units for I (cm^2 )  
  
# calculate the covariance matrix manually  
# Assuming homoscedasticity and variance of error term is sigma^2  
sigma2 <- SSR / (length(I\_observed) - 2) # 2 parameters estimated  
X <- cbind(sqrt(t), t)  
cov\_matrix <- sigma2 \* solve(t(X) %\*% X)  
  
# Print the SSR and covariance matrix  
SSR

## [1] 2.676812e-05

cov\_matrix

## t  
## 3.024404e-05 -3.457587e-05  
## t -3.457587e-05 4.363683e-05

Check again with lm function

fit <- lm(I ~ sqrt\_t + t - 1, data = data) # Make sure data has sqrt\_t and t  
  
# The sum of squared residuals can be obtained from the model object  
SSR <- sum(resid(fit)^2)  
  
# The covariance matrix is obtained from the summary of the model object  
cov\_matrix <- vcov(fit)  
  
# Print the SSR and covariance matrix  
SSR

## [1] 2.676812e-05

cov\_matrix

## sqrt\_t t  
## sqrt\_t 3.024404e-05 -3.457587e-05  
## t -3.457587e-05 4.363683e-05

***part 2.7***

# Number of observations  
n <- length(residuals)  
# Number of parameters estimated, which is 2 in this case (S and Ks)  
p <- 2  
  
# Extract the standard errors from the covariance matrix  
standard\_errors <- sqrt(diag(cov\_matrix))  
# or I could do sigma\_squared <- sum(residuals^2) / (n - p)  
# Assuming normality, the critical value for a 95% confidence interval is approximately 1.96  
critical\_value <- qt(0.975, df=n-p) # More accurate using t-distribution with n-p degrees of freedom  
  
# Compute the confidence intervals for each parameter  
confidence\_intervals <- data.frame(  
 lower = theta\_hat - critical\_value \* standard\_errors,  
 upper = theta\_hat + critical\_value \* standard\_errors  
)

## Warning in theta\_hat - critical\_value \* standard\_errors: longer object length  
## is not a multiple of shorter object length

## Warning in theta\_hat + critical\_value \* standard\_errors: longer object length  
## is not a multiple of shorter object length

# Print the confidence intervals  
print(confidence\_intervals)

## lower upper  
## 1 1.9858632 2.014137  
## 2 3.9830192 4.016981  
## 3 -3.0141368 -2.985863  
## 4 -7.0169808 -6.983019  
## 5 0.9858632 1.014137