ILLINOIS INSTITUTE OF TECHNOLOGY

MATHEMATICAL MODELING

MATH 522 - PROJECT: PHARMACOKINETICS AND MODEL FITTING

Name: Sai Nivas Rangaraju
College: Illinois Institute of Technology
Date: Nov 19, 2023

Summary
Parameter fitting and Two compartment model.

0.1 Introduction

Physiological systems refer to the complex and organized networks of organs, tissues, cells, and molecules within an organism that work together to perform specific functions necessary for the maintenance of life.

Physiological systems, particularly those involving drug distribution and elimination, are often modeled to gain insights into their behavior and optimize therapeutic strategies. In this project, we focus on a two-compartment model, a common approach in Pharmacokinetics, where a physiological system is divided into two distinct compartments to describe the distribution and elimination of a drug.

0.1.1 Problem Statement

Part 1: Parameter Estimation in a Two-Compartment Model.

The concentration in the second compartment (plasma) of our two-compartment model is characterized by a functional form involving exponentials. The concentration, denoted as $y(t)$, is given by the equation:

$$y(t) = b(1) * e^{b(2)*t} + b(3) * e^{b(4)*t}, t \geq 0$$

Here b(j), j = 1,2,3,4, are unknown parameters that we seek to estimate.

Objective

The primary objective of Part 1 is to explore the concept of exponential peeling and implement it for parameter estimation in a two-compartment model. The model describes the concentration in the second compartment (plasma) as a sum of exponentials, and the unknown parameters ($b(j)$) need to be estimated.

Part 2: Estimating Rate Constants for a Two Compartment Model.

In this Part 2, the focus shifts to estimating rate constants for a two-compartment model that describes the dynamics of drug distribution between the central and tissue compartments.

The central compartment, consisting of blood and extracellular water, is rapidly diffused with the drug. The second compartment, known as the tissue compartment, contains tissues that equilibriate more slowly with the drug. If x1 is the concentration of drug in the blood and x2 is its concentration in the tissue, the compartment model is described by the following system:

$$x_1' = -(k_{01} + k_{21})x_1 + k_{12}x_2$$
$$x_2' = k_{21}x_1 - k_{12}x_2$$

where k_{01}, k_{21}, k_{12} represents the rate constants.

Objective

Estimating rate constants for a two-compartment model that describes the dynamics of drug distribution between the central and tissue compartments. This part focuses on the implementation of exponential peeling, model fitting and the concept of Eigen values and Eigen vectors.

0.1.2 Mathematical Methods used

Part 1

In solving the part1, exponential peeling, model fitting and optimization concepts are used. The solution and the method goes as:

Exponential Peeling:

We got $y(t) = b(1) * e^{b(2)*t} + b(3) * e^{b(4)*t}, t \geq 0$. This is a hard problem to fit a linear model to it, as we can't take *log* for them and then fit the model. Hence, we use the concept of *exponential peeling.*

In exponential peeling, we make an assumption that $b(4) < b(3) < 0$. We then rewrite the equation as:

$y(t) = c_1 * e^{\lambda_1 * t} + c_2 * e^{\lambda_2 * t}$ —-(1)

$$=> y(t) = (c_1 * e^{(\lambda_1 - \lambda_2) * t} + c_2) * e^{\lambda_2 * t}$$

where, $c_1 = b(1), c_2 = b(2), \lambda_1 = b(3), \lambda_2 = b(4)$ and $\lambda_2 << \lambda_1$.

When $t >> 1$, then $e^{(\lambda_1 - \lambda_2) * t} \approx e^{(\lambda_1) * t}$, then the equation y(t) becomes:

$$y(t) = (c_1 * e^{(\lambda_1) * t})$$

Now, we can apply *log* to the function on both sides

$log(y) = log(c_1) + \lambda_1 * t$ —- (2)

After finding the values of λ_1 and c_1, we have:

$$y^*(t) = y(t) - c_1 * e^{(\lambda_1) * t}$$

$$=> y^*(t) \approx c_2 * e^{\lambda_2 * t}$$

The y^* represents the residuals left after $c_1 * e^{(\lambda_1) * t}$ is captured. This gives us the residuals caused by the 2nd term, $c_2 * e^{\lambda_2 * t}$. Then we will repeat the process of applying *log* on both sides and then fit a new linear model to obtain the parameters λ_2 and c_2 as: $log(y^*(t)) = log(c_2) + \lambda_2 * t$ —-(3)

Linear Model fitting:

We now have a linear equation of the form $f(t) = c + m * t$, where $f(t) = y, c = log(c_1), m = \lambda_1$.

Now we can proceed to fit a linear model to the equation using the data provided for a selected time .

The linear model adjusts the values of c(intercept) and m(slope) to obtain the best fit. By best fit, the model tries to minimize the Sum of Squared errors metric.

From the model, we can obtain the values of λ_1 and c_1.

Now we find the y^*, which leads us to fitting the 2nd linear model to find the values of λ_2 and c_2.

Optimization:

Many mathematical and programming softwares provide packages that provide us with optimization functions that help us obtain optimal values for parameters for a given equation. We will be using one such programming software R-studio, where we code in R language and use a optimization function to obtain the optimal values for the parameters: λ_1, c_1, λ_2 and c_2.

Part 2

For part 2, we have two functions $x_1'(t)$ and $x_2'(t)$. We represent this as:

$x' = K * x$ —–(4)

where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} = \begin{bmatrix} -(k_{01} + k_{21}) & k_{12} \\ k_{21} & -k_{12} \end{bmatrix}$.

Eigen Values and Eigen Vectors:

In this part, we use the concepts of Eigen values as well as Eigen vectors, as we are using matrices.

Given equation $x^*(t_k) = x_1^*(t_k)\mathbf{i} + x_2^*(t_k)\mathbf{j}$ and assumed Eigen values as λ_1 and λ_2, the corresponding Eigen vectors are given as:$v_1 = \begin{bmatrix} v_{11} \\ v_{21} \end{bmatrix}$ and $v_2 = \begin{bmatrix} v_{12} \\ v_{22} \end{bmatrix}$ and the solution for the equation (4) is: $x(t) = \alpha e^{\lambda_1 t} v_1 + \beta e^{\lambda_2 t} v_2$, where α and β are non-zero values.

Exponential Peeling:

Given $\lambda_2 < \lambda_1 < 0$, hence we can rewrite the above equation as:

$$x(t) = e^{\lambda_1 t}(\alpha v_1 + \beta e^{(\lambda_2 - \lambda_1)t} v_2)$$

since $\lambda_2 < \lambda_1 < 0$, we can consider that for some time t>>0, the $e^{(\lambda_2 - \lambda_1)t} \approx 0$.

Hence we deduce the equation as: $x(t) \approx \alpha e^{\lambda_1 t} v_1$ —-(5).

Linear Model Fitting:

Applying *log* to (5) gives us: $log(x) = \lambda_1 t + log(\alpha v_1)$ —-(6).

We obtain from (6): $log(x_1) = \lambda_1 t + log(\alpha v_{11})$ and $log(x_2) = \lambda_1 t + log(\alpha v_{21})$ —-(7).

Now, we consider the residuals $x^*(t) = x(t) - \alpha e^{\lambda_1 t} v_1 = \beta e^{\lambda_2 t} v_2$ —-(8)

This gives us: $x_1^*(t) = \beta e^{\lambda_2 t} v_{12}$, $x_2^*(t) = \beta e^{\lambda_2 t} v_{22}$ —-(9).

Applying *log* to the above equation, we get: $log(x_1^*) = \lambda_2 t + log(\beta v_{12})$ and $log(x_2^*) = \lambda_2 t + log(\beta v_{22})$ —-(10).

Properties of Quadratic Equations:

We will also use the below properties of quadratic equations. Let's suppose we have a quadratic equation: $ax^2+bx+c = 0$ and x_1 and x_2 are the roots of the equation. Then: Sum of roots is given as: $x_1 + x_2 = \frac{-b}{a}$ and the product of roots is given as: $x_1.x_2 = \frac{c}{a}$.

0.2 Technical Section: Results

I have used **R** language for the coding part of the problem, especially for model fitting.

0.2.1 Part 1

1.a. From (2) we got: $log(y) = log(c_1) + \lambda_1 * t$.

Note: *I have used the time split $t \geq 4$*

Using the *lm()* function in R, using the data given, I have fit the linear model to the above equation for the data with time≥4 . The linear model gave us the values of $\lambda_1 = -0.11$ and $c_1 = 5.33$.

Now from (3), we have: $log(y^*(t)) = log(c_2) + \lambda_2 * t$.

Using the *lm()* again, using the data with time≤4, I have fit the linear model. The values obtained by the linear model's output are: $\lambda_2 = -1.31$ and $c_2 = 5.55$.

Sum of Squared Errors: The Sum of Squared Errors for the estimated vs actual values of y is given as:

$$SSE_{estimated} = 3.6562$$

1.b. The graph for the actual data points vs the estimated values using parameters of exponential peeling is as shown in figure 1(in the Graphs and figures section). It is clear from the figure that the fit is reasonable and estimates the actual values almost correctly.

1.c. Optimization

In **R** language, we are provided with a function called *optim()* which calculates optimal parameter values to obtain the optimal value of interest.

I have used the optim function here to get the parameter values to minimize the Sum of Squared Errors and the values the function provided are given as: $\lambda_1 = -0.104$, $c_1 = 5.13$, $\lambda_2 = -1.08$ and $c_2 = 4.94$.

The values we obtained are almost similar to the values we obtained by the optimization function.

Sum of Squared Errors: The sum of squared errors for the actual data points vs the estimated values using optimal parameters is given as:

$$SSE_{optimal} = 0.9279$$

The figure 2 shows the graph of actual values vs estimated values vs values obtained by using optimal parameters.

0.2.2 Part 2

1. From (4), we have $x' = K * x$. Let $x = w * e^{\lambda * t}$, then $x' = w * \lambda * e^{\lambda * t}$

$=> w * \lambda * e^{\lambda * t} = K * w * e^{\lambda * t} => (K - \lambda I)w = 0$.

For the non-trivial solution of $w \neq 0$, we have —$K - \lambda I$— $= 0$.

Now using the entries of K and I(identity matrix), we get:

$$\det \begin{bmatrix} -(k_{01} + k_{21} + \lambda) & k_{12} \\ k_{21} & -(k_{12} + \lambda) \end{bmatrix} = 0$$

$$=> (k_{01} + k_{21} + \lambda) * (k_{12} + \lambda) - k_{21} * k_{12} = 0$$

$$=> \lambda^2 + (k_{01} + k_{21} + k_{12}) * \lambda + k_{01} * k_{12} = 0$$

Now by solving the above, we obtain the values of the eigen values: λ_1 and λ_2.

$$\lambda_{1,2} = \frac{(-(k_{01} + k_{21} + k_{12}) \pm \sqrt{(k_{01} + k_{21} + k_{12})^2 - 4 * (k_{01} * k_{12})})}{2}$$

here a = 1, b = $k_{01} + k_{21} + k_{12}$, c = $k_{01} * k_{12}$.

Since all the rates are positives, $b^2 - 4ac$ is:

$$(k_{01} + k_{21} + k_{12})^2 - 4 * (k_{01} * k_{12})$$

$$=> k_{01}^2 + k_{21}^2 + k_{12}^2 + 2k_{01}k_{21} + 2k_{12}k_{21} - 2k_{01}k_{12}$$

This is always greater than 0. Hence λ_1 and λ_2 are real and distinct.

Now if we compare b and $\sqrt{b^2 - 4ac}$, we have: b^2 and $b^2 - 4ac$. Hence $b^2 > \sqrt{b^2 - 4ac}$.

Let $\lambda_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ and $\lambda_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$, hence $\lambda_2 < 0$ and from above we understand that $\lambda_1 < 0$ since $-b > \sqrt{b^2 - 4ac}$.

a. Hence, λ_1 and λ_2 are real, distinct, and negative.

b. From the properties of quadratic equation, we know that:

$\lambda_1 + \lambda_2 = \frac{-b}{a} = -(k_{01} + k_{21} + k_{12})$ and $\lambda_1 * \lambda_2 = \frac{c}{a} = k_{01} * k_{12}$.

2. From equation (7), it is clear that the graphs for $ln x_1(t)$ and $ln x_2(t)$ are straight lines with slope λ_1 and intercepts $ln \alpha v_{11}$ and $ln \alpha v_{21}$, respectively. We now fit two linear models to these two equations and obtain the values of λ_1, αv_{11} and αv_{21}.

Note: The λ_1 value obtained from both the linear models are almost similar, we can hence take average of both the values from both models.

Now, after obtaining the values of λ_1, αv_{11} and αv_{21}, we can get the residual data as:

$$x^*(t_n) = x(t_n) - v_1^{(\alpha)} e^{(\lambda_1 t_n)} \approx \beta e^{\lambda_2 t} v_2$$

where estimates αv_1 are denoted by $v_1^{(\alpha)}$.

Now apply *log* on the equation, we get:

$$log(x^*(t_n)) = \lambda_2 * t + log(\beta * v_2)$$

From this we obtain: $log(x_1) = \lambda_2 t + log(\beta v_{12})$ and $log(x_2) = \lambda_2 t + log(\beta v_{22})$.

These are linear equations too involving the log of residuals vs t. We now fit two linear models to these two equations and obtain the values of λ_2, βv_{12} and βv_{22}. Figure 4 shows the plot for $log(x_1)$ and $log(x_2)$.

3. Since, λ_1 and λ_2 are the two Eigen values and $\bar{v}$ represents the Eigen vectors, and let $\bar{A}$ be the diagonal matrix of Eigen values, then we've from diagonalization concept: $\bar{A} = \bar{v}^{-1} \bar{K} \bar{v} => \bar{K}\bar{v} = \bar{v}\bar{A}$ —- (11)

where $\bar{K} = \begin{bmatrix} -(k_{01} + k_{21}) & k_{12} \\ k_{21} & -k_{12} \end{bmatrix}$, $\bar{v} = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$ and $\bar{A} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$.

4. Given the estimates of $\hat{K_{ij}}$, representing the entries of K and the estimated values of the Eigen values as: $\hat{\lambda_1}$ and $\hat{\lambda_2}$. We have considered from equation 4 that

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} = \begin{bmatrix} -(k_{01}+k_{21}) & k_{12} \\ k_{21} & -k_{12} \end{bmatrix}$$

We get, $\hat{K_{11}} = -(\hat{k_{01}} + \hat{k_{21}})$, $\hat{K_{12}} = \hat{k_{12}}$, $\hat{K_{21}} = \hat{k_{21}}$ and $\hat{K_{22}} = -\hat{k_{12}}$.

From **1.b** we get

$$\hat{\lambda_1} + \hat{\lambda_2} = \frac{-b}{a} = -(\hat{k_{01}} + \hat{k_{21}} + \hat{k_{12}})$$

since we have the values of the estimates of $\hat{\lambda_1}$, $\hat{\lambda_2}$ and the values of $\hat{K_{ij}}$:

$\hat{k_{01}} = -(\hat{\lambda_1} + \hat{\lambda_2} + \hat{k_{21}} + \hat{k_{12}}) = -(\hat{\lambda_1} + \hat{\lambda_2} + \hat{K_{21}} + \hat{K_{12}})$ — (12)

5. As mentioned earlier, I used **R** language to code. Given the compartment concentration measurements, we first perform the exponential peeling, with the timesplit considered at t = 50 and get to the equation (7): $log(x_1) = \lambda_1 t + log(\alpha v_{11})$ and $log(x_2) = \lambda_1 t + log(\alpha v_{21})$. We fit linear models to these equations and obtain the values of $\lambda_1 = -0.0174$, $\alpha v_{11} = 0.2098$ and $\alpha v_{21} = 0.2629$.

Note: *The values of λ_1 obtained from both the linear models is approximately same, I have taken average of the values, as a general way to normalize the value.*

Now from (10), we get $log(x_1^*) = \lambda_2 t + log(\beta v_{12})$ and $log(x_2^*) = \lambda_2 t + log(\beta v_{22})$. Fitting linear models to the linear equations, we get the values of $\lambda_2 = -0.1194$, $\alpha v_{21} = 0.4626$ and $\alpha v_{22} = -0.4626$.

Note: *For the linear model 2, for the time < timesplit (50 in my case), the 2nd linear model can't be fitted as there are negative values to which log is applied and these yields NA values. Hence, we discard the linear model and take λ_2 value obtained from 1st linear model and the value of αv_{21} value is obtained from the initial condition which gives us: $\alpha v_{22} = -\alpha v_{21}$.*

From (11), we get that $\bar{K}\bar{v} = \bar{v}\bar{A}$. From this we obtain the following system of linear equations, obtained as:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$K_{11}v_{11} + K_{12}v_{21} = v_{11}\lambda_1 => (\alpha v_{11})K_{11} + (\alpha v_{21})K_{12} = (\alpha v_{11})\lambda_1,$$

$$K_{11}v_{12} + K_{12}v_{22} = v_{12}\lambda_2 => (\beta v_{12})K_{11} + (\beta v_{22})K_{12} = (\beta v_{12})\lambda_2,$$

$$K_{21}v_{11} + K_{22}v_{21} = v_{21}\lambda_1 => (\alpha v_{11})K_{21} + (\alpha v_{21})K_{22} = (\alpha v_{21})\lambda_1,$$

$$K_{21}v_{12} + K_{22}v_{22} = v_{22}\lambda_2 => (\beta v_{12})K_{21} + (\beta v_{22})K_{22} = (\beta v_{22})\lambda_2.$$

Now from above we can substitute the obtained values of $\lambda_1 = -0.0174$, $\alpha v_{21} = 0.2098$ and $\alpha v_{21} = 0.2629$, $\lambda_2 = -0.1194$, $\alpha v_{21} = 0.4626$ and $\alpha v_{22} = -0.4626$ and solving the system of linear equations gives us the values of $K_{11} = -0.0738, K_{12} = 0.0452, K_{21} = 0.056$ and $K_{22} = -0.624$.

Now from (12), we know that:

$$k_{01} = -(\lambda_1 + \lambda_2 + K_{21} + K_{12}), k_{12} = -K_{22}, k_{21} = K_{21}$$

$$=> k_{01} = 0.0356, k_{12} = 0.624, k_{21} = 0.056$$

The graph for the estimated values of x(x_1 and x_2)against time(t) is as shown in the figure 3. From the graph, we can say that the estimated values are almost similar to the actual values and the fit is a reasonable one.

Sum of Squared Errors: The sum of squared errors between the estimated values and actual values for x_1 and x_2 are given as:

$$SSE_{x_1} = 8.7041 * 10^{-5}, SSE_{x_2} = 0.0072$$

Note: *The plot line for estimated x_2 could not catch the real values exactly. This is due to the negative values of x_2 for which we directly took the value of βv_{22} from the initial condition. However, that is the best possible fit.*

0.3 Graphs or Figures:

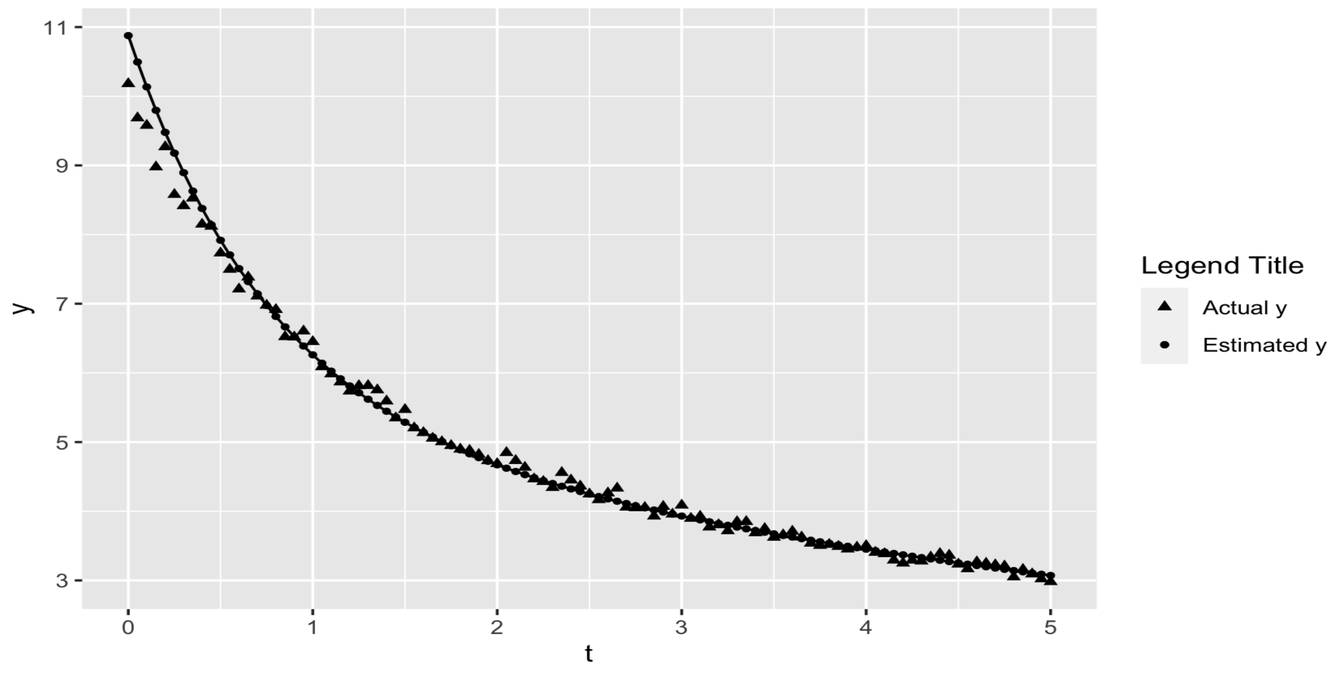

Figure 1: Estimated y vs Actual y

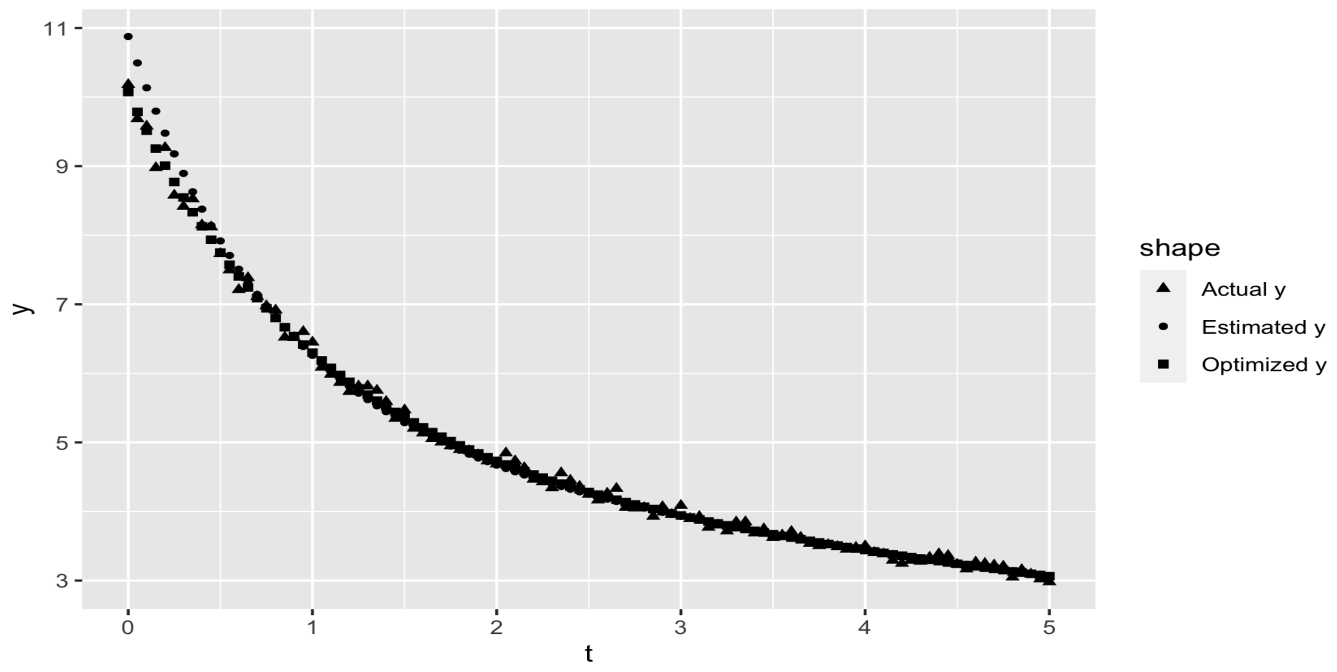

Figure 2: Actual y vs Exponential peeling's y vs Optimized parameters' y

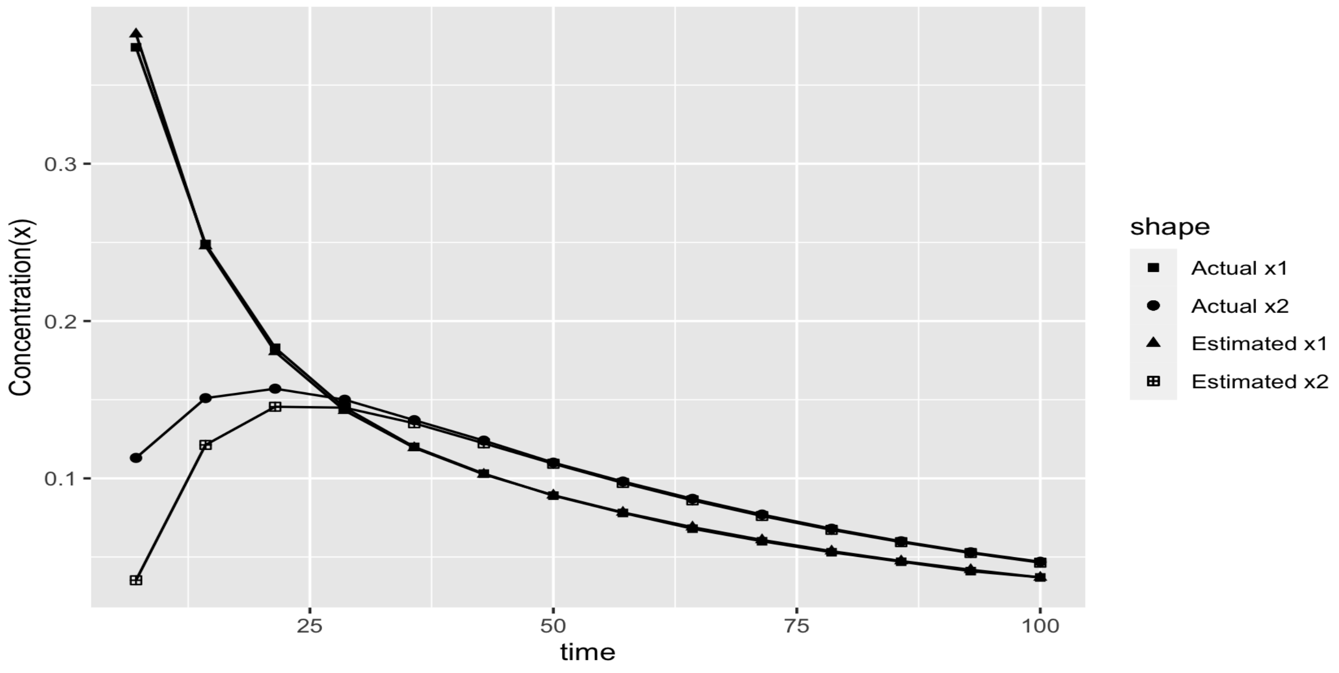

Figure 3: Estimated concentrations vs Actual concentrations

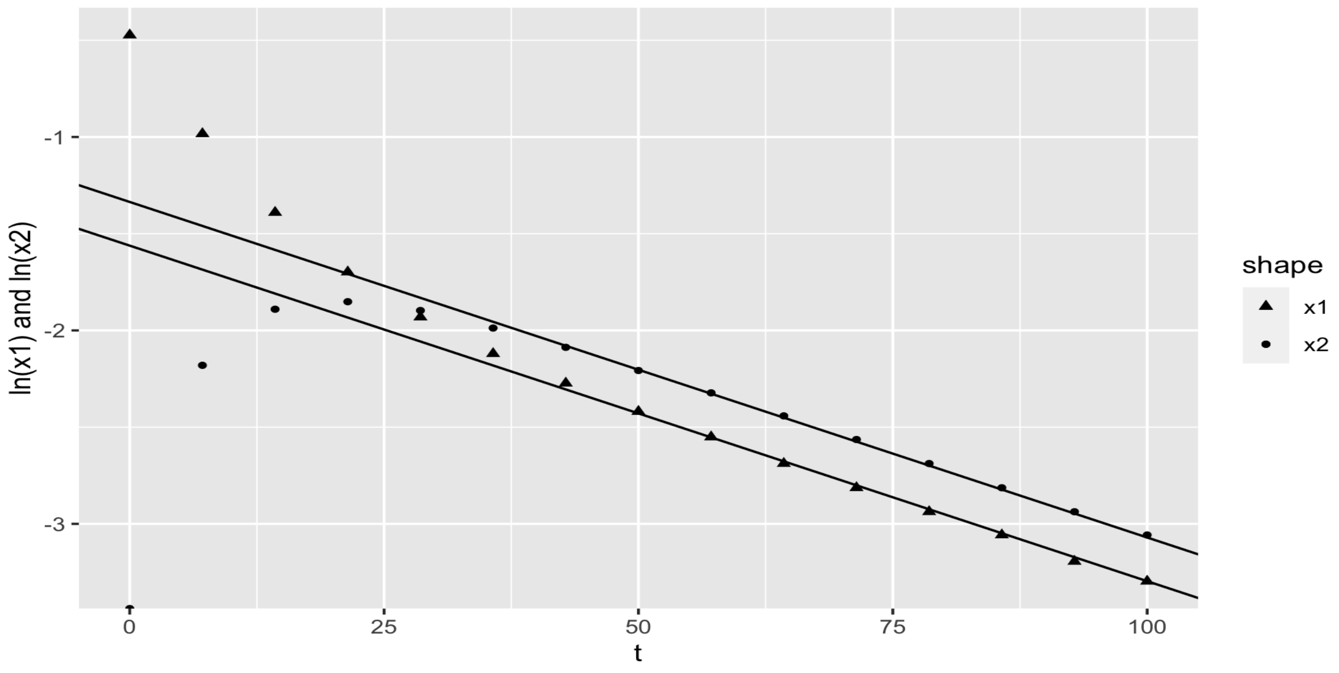

Figure 4: log(x1) and log(x2) vs t

0.4 List of Reference(s):

1. PEEL - A Program to Perform Exponential ”Peeling” (Fitting) On-Line by Lyle B. Smith, Computation Group, Stanford Linear Accelerator Center.
2. Journal of Theoretical Biology by D. Van Liew (1967).

0.5 Appendix

~~~
#Part1

##Loading dataset into dataframe
```{r}
library(readxl)
data <- read_excel(path = "/Users/sainivasrangaraju/Desktop/MMData.xlsx")
~~~

##New data frame with t and ln_y values
```{r}
lny_full_data <- data.frame(t = data$t,ln_y = log(data$y))
head(lny_full_data)

#Uncomment and run the below command to view completed data in lny_full_data
#lny_full_data
```

##Time selected for exponential peeling(t>4)
```{r}
time_selected <- which(data$t>4)
```

##Fitting a linear model
```{r}
lin1_model <- lm(ln_y~., lny_full_data, subset = time_selected)
lin1_model
```

##Extracting coefficients of the model and thus obtaining c1 and l1(lambda1)
```{r}
c1 <- exp(lin1_model$coefficients[1])
c1

l1 <- lin1_model$coefficients[2]
l1
```
##Finding y-c1*e^(-l1*t), the residual
```{r}
y_new <- data$y-c1*exp(l1*data$t)
head(y_new)

#Uncomment and run the below command to view completed data in y_new
#y_new
```

##Creating new dataframe with the log(residual_y)
```{r}
final_data <- data.frame(t = data$t,ln_y_res = log(y_new))
head(final_data)
```

##Fitting linear model to the residual log values
```{r}
lin2_model <- lm(ln_y_res~., final_data, subset = -time_selected)
lin2_model
```
```
```

~~~
##Extracting coefficients of the model and thus obtaining c2 and l2(lambda2)
```{r}
c2 <- exp(lin2_model$coefficients[1])
c2

l2 <- lin2_model$coefficients[2]
l2
```

##Creating new dataframe with final estimated values of y wrt t
```{r}
estimated_dataframe <- data.frame(y = c1*exp(l1*data$t)+c2*exp(l2*data$t),
                                  t = data$t)
head(estimated_dataframe)

#Uncomment and run the below command to view the  data in estimated_dataframe
#estimated_dataframe
```

##Graph for estimated vs actual y values
```{r}
library(ggplot2)
ggplot()+
  geom_point(data = data,aes(t,y, shape = "Actual y"))+
  geom_point(data = estimated_dataframe, aes(t,y, shape = "Estimated y"))+
  geom_line(data = estimated_dataframe, aes(t,y))+
  ggtitle("Estimated y vs Actual y")+
  labs(shape = "Legend Title")+
  scale_shape_manual(values = c("Actual y" = 17, "Estimated y" = 20))
```

##Sum of Squared Errors for the estimated vs actual
```{r}
SR <- (estimated_dataframe$y-data$y)^2
SSR <- sum(SR)
cat("Sum of Squared Errors for the estimated vs actual is ",SSR)
```

##Using Optimization function optim() to find the c1, l1, c2 and l2
```{r}
objective_function <- function(x) {
  SE = (data$y-x[1]*exp(x[2]*data$t)-x[3]*exp(x[4]*data$t))^2
  SSE = sum(SE)
  return(SSE)
}

# Initial guess
initial_guess <- c(0,0,0,0)

# Call optim
result <- optim(par = initial_guess, fn = objective_function, method = "BFGS")

# Display results
cat("\nMinimum found at:")
cat("\n\tc1 =", result$par[1], "\n\tl1 =", result$par[2],
    "\n\tc2 =", result$par[3], "\n\tl2 =", result$par[4])
~~~

```
cat("\nThe minimum SSE obtained is =", result$value, "\n")
‘‘‘
##Creating dataframe for y values estimated by optimized paramaters
‘‘‘{r}
optimized_dataframe <- data.frame(t = data$t,y = result$par[1]*exp(result$par[2]*data$t)+result$par[3]*exp(
head(optimized_dataframe)
‘‘‘

##Sum of Squared Errors for the actual vs y obtained by optimized parameters
‘‘‘{r}
SE <- (optimized_dataframe$y-data$y)^2
SSE <- sum(SE)
cat("Sum of Squared Errors for the optimized vs actual is ",SSE)
‘‘‘


##Graph for actual y vs estimated y vs y obtained by optimized parameters
‘‘‘{r}
ggplot()+
  geom_point(data = data,aes(t,y, shape = "Actual y"))+
  geom_point(data = estimated_dataframe, aes(t,y, shape = "Estimated y"))+
  geom_point(data = optimized_dataframe, aes(t,y, shape="Optimized y"))+
  scale_shape_manual(values = c("Actual y" = 17,
                          "Estimated y" = 20,"Optimized y" = 15))+
  ggtitle("Original y vs Exponential peeling’s y vs Optimized parameters’ y")
‘‘‘

##Comparing the exponential peeling result vs optimized parameters result
The optimization function gave us the parameters values as: c1 = 5.132154,
l1 = -0.1046885, c2 = 4.941015 and l2 = -1.080367.
These values are almost similar to the values obtained by exponential
peeling, the values are c1 = 5.328934, l1 = -0.1107134,
c2 = 4.447866 and l2 = -1.12763

It is also clear from the graph that the y-values(points) obtained by using
exponential peeling(points in purple) and by using optimized parameters
(points in cyan) overlap almost perfectly with the original y values(points in
orange)


#Part 2

##Loading dataset into dataframe
‘‘‘{r}
data <- readxl::read_excel("/Users/sainivasrangaraju/Desktop/MM Part2.xlsx")
head(data)

#Uncomment the below code to view full data
#data
‘‘‘

##Time selected for exponential peeling(t>50)
‘‘‘{r}
time_split <- which(data$time > 50)
‘‘‘

##New data frame with t, ln(x1) and ln(x2) values
```

```
‘‘‘{r}
log_data <- data.frame(t = data$time, ln_x1 = log(data$x1), ln_x2 = log(data$x2))
log_data
‘‘‘

##Fitting linear model to ln(x1) and t
‘‘‘{r}
lin1_model <- lm(ln_x1~t, log_data, subset = time_split)
lin1_model
‘‘‘

##Fitting linear model to ln(x2) and t
‘‘‘{r}
lin2_model <- lm(ln_x2~t, log_data, subset = time_split)
lin2_model
‘‘‘

##Obtaining lambda1, alpha*v11 and alpha*v21
‘‘‘{r}
lambda1 <- (lin1_model$coefficients[2]+lin2_model$coefficients[2])/2
lambda1
‘‘‘

‘‘‘{r}
c1 <- exp(lin1_model$coefficients[1])
#c1 is alpha*v11
c1
‘‘‘

‘‘‘{r}
c2 <- exp(lin2_model$coefficients[1])
#c2 is alpha*v21
c2
‘‘‘

##Plotting the linear models with log(x1) and log(x2)
‘‘‘{r}
library(ggplot2)
ggplot(log_data, aes(x = t))+
  geom_point(aes(y = ln_x1, shape = "x1"))+
  geom_point(aes(y = ln_x2, shape = "x2"))+
  geom_abline(intercept = lin1_model$coefficients[1], slope = lambda1)+
  geom_abline(intercept = lin2_model$coefficients[1], slope = lambda1)+
  scale_shape_manual(values = c("x1" = 17, "x2" = 20))+
  labs(title = "x1 and x2 vs t",y="ln(x1) and ln(x2)", x = "t")
‘‘‘

##Calculating the residuals of x1 and x2
‘‘‘{r}
new_x1 <- data$x1 - c1*exp(lambda1*data$time)
new_x2 <- data$x2 - c2*exp(lambda1*data$time)
‘‘‘

##Creating dataframe for log of the residuals of x1 and x2
‘‘‘{r}
new_log_data<-data.frame(t = data$time,ln_x1 = log(new_x1),ln_x2 = log(new_x2))
new_log_data
```

```
```

##Fitting linear model to the ln(x1*)
```{r}
lin3_model <- lm(ln_x1~t, new_log_data, subset = -time_split)
lin3_model
```

##Obtaining lamda2
```{r}
lambda2 <- as.numeric(lin3_model$coefficients[2])
lambda2
```

##Obtaining beta*v12
```{r}
c3 <- as.numeric(exp(lin3_model$coefficients[1]))
c3
```

##Taking beta*v22 = -(beta*v12)(from Initial conditions)
```{r}
#Here we can't fit the linear model to the log of x2* as the x2* values are
#negative and the log will be not defined. To avoid this we skipped fitting
#linear model to the log(x2*) and calculated the value of beta*v22 from the
#initial condition i.e; the first entry in the data, which gives us:
#beta*v22 = -(beta*v12)
c4 <- -c3
c4
```

##Calculating estimated values
```{r}
#time[-1] excludes the first entry i.e; the initial condition
estimated_x1 <- c1*exp(lambda1*data$time[-1])+c3*exp(lambda2*data$time[-1])
estimated_x2 <- c2*exp(lambda1*data$time[-1])+c4*exp(lambda2*data$time[-1])
```

##Creating dataframe with estimated values
```{r}
estimated<-data.frame(t=data$time[-1],x1=estimated_x1,x2=estimated_x2)
```

##Calculating Sum of Squared errors for both x1 and x2
```{r}
x1_sse <- sum((data$x1[-1]-estimated$x1)^2)
x1_sse
```
```{r}
x2_sse <- sum((data$x2[-1]-estimated$x2)^2)
x2_sse
```

##Plotting the estimated vs actual values of x1 and x2
```{r}
ggplot()+
geom_point(data,mapping=aes(x =time,y = x1, shape="Actual x1"))+
geom_point(estimated,mapping=aes(x =t,y = x1, shape="Estimated x1"))+
geom_point(data,mapping=aes(x =time,y = x2, shape="Actual x2"))+
geom_point(estimated,mapping=aes(x =t,y = x2, shape="Estimated x2"))+
geom_line(data,mapping=aes(x =time,y = x1))+
geom_line(estimated,mapping=aes(x =t,y = x1))+
geom_line(data,mapping=aes(x =time,y = x2))+
geom_line(estimated,mapping=aes(x =t,y = x2))+
scale_shape_manual(values = c("Actual x1" = 15, "Estimated x1" = 17,
                                "Actual x2" = 19, "Estimated x2" = 12))+
ggtitle("Estimated concentrations vs Actual concentrations")+
labs(x = "time", y = "Concentration(x)")
‘‘‘
```