MTH686A Non Linear Regression

Project By - Alok Yadav 210102 yalok21@iitk.ac.in

Introduction

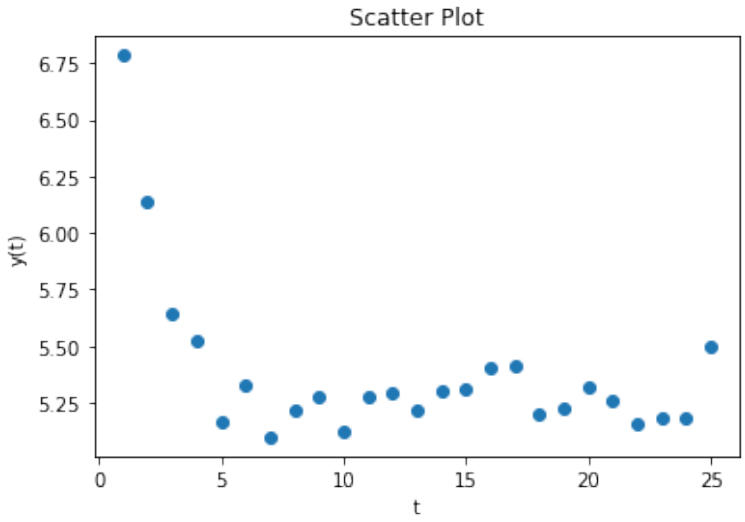
We were given 25 observations on the variable y. The model to be fitted is given by:

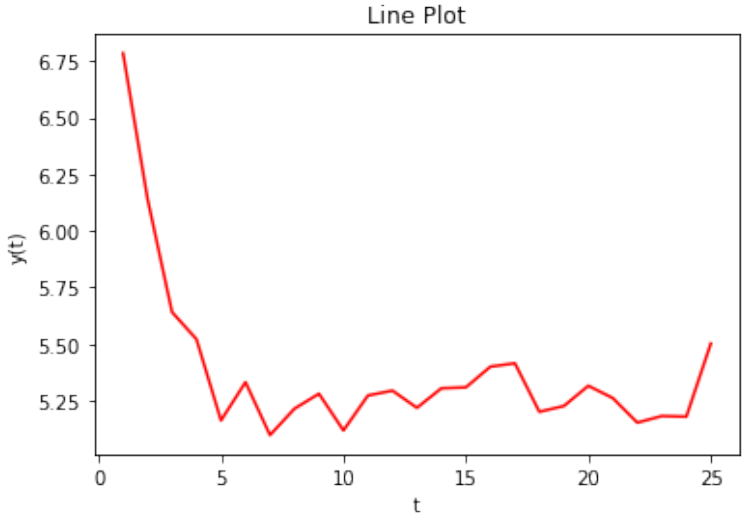
$$y_t = \alpha_1 + \alpha_2 \exp(\beta t) + \epsilon_t$$

Here, ϵ_t is a sequence of i.i.d. random variables with mean zero and finite variance.

Part(1)

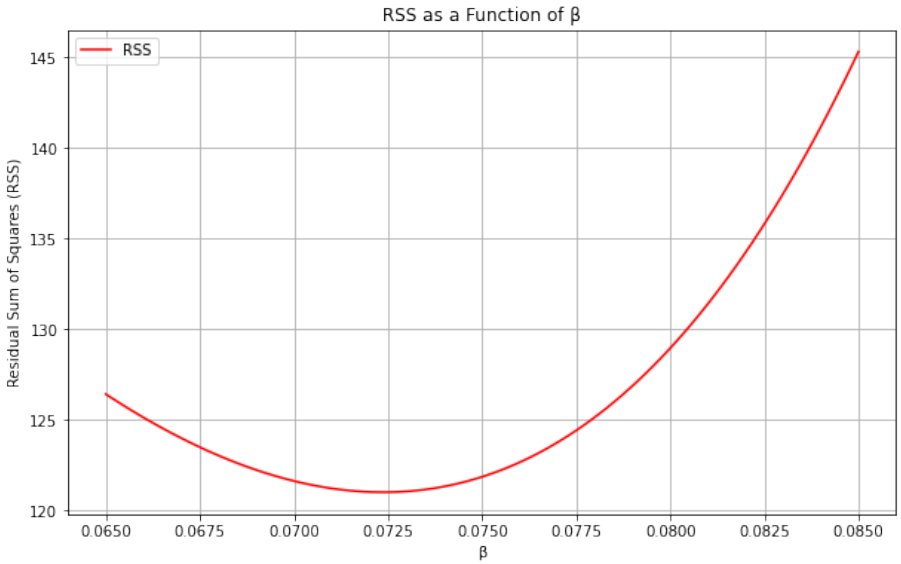
In this plot, the x-axis represents the variable t, and the y-axis represents $y(t)$. The data appears to have a non-linear trend.





Part(2)

Define residuals as $y - \hat{y}$.



Part(3)

The Gauss-Newton method is an iterative optimization technique used for solving non-linear least squares problems. Given a model function $f(\theta)$ and a set of parameters $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$, the goal is to minimize the sum of squared residuals:

$$Q(\theta) = \sum_{i=1}^{N} [y_i - f_i(\theta)]^2$$

Here, N is the number of observations, y_i is the observed value, and $f_i(\theta)$ is the predicted value from the model.

Thus applying Taylor series approximation

$$Q(\theta)) \approx Q(\theta_0) + \mathbf{g}(\theta_0)(\theta - \theta_0) + 1/2 * \mathbf{H}(\theta_0)(\theta - \theta_0)^T(\theta - \theta_0)$$

The Gauss-Newton method iteratively updates the parameter estimates to minimize the sum of squared residuals. The update at each iteration is given by:

$$\theta_{a+1} = \theta_a + \delta_a$$

where δ_a is the solution to the linearized least squares problem:

$$\delta_a = -(\mathbf{H}_a)^{-1} \cdot \mathbf{g}_a$$

Here, $\mathbf{H}_a$ is the Hessian matrix, whose elements are the partial derivatives of the model function with respect to the parameters, evaluated at the current estimates θ_a. $\mathbf{r}_a$ is the vector of residuals.

The Hessian matrix $\mathbf{H}_a$ is defined as:

$$\mathbf{H}_a = \begin{bmatrix} \frac{\partial^2 Q_1(\theta)}{\partial {\theta_1}^2} & \frac{\partial^2 Q_1(\theta)}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 Q_1(\theta)}{\partial \theta_1 \partial \theta_n} \\ \frac{\partial^2 Q_2(\theta)}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 Q_2(\theta)}{\partial {\theta_2}^2} & \cdots & \frac{\partial^2 Q_2(\theta)}{\partial \theta_2 \partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 Q_N(\theta)}{\partial \theta_n \partial \theta_1} & \frac{\partial^2 Q_N(\theta)}{\partial \theta_n} \partial \theta_2 & \cdots & \frac{\partial^2 Q_N(\theta)}{\partial {\theta_n}^2} \end{bmatrix}$$

The iterative process continues until a convergence criterion is met, such as $|\theta_{a+1} - \theta_a| < 0.0005$. After iterations, the Gauss-Newton method has converged.

Final Estimated Parameters:

- $\hat{\alpha}_1$: 5.254293777650497
- $\hat{\alpha}_2$: 3.1153066828267457
- $\hat{\beta}$: -0.685408710728477

Part(4)

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (\text{residuals }_i)^2$$

Estimated σ^2: 0.010202

Part(5)

The associated confidence intervals based on the Fisher information matrix:

$$\text{CI}_{\theta_i} = \left(\hat{\theta}_i - z_{\alpha/2} \sqrt{\left[\mathbf{I}(\hat{\theta})^{-1} \right]_{ii}}, \hat{\theta}_i + z_{\alpha/2} \sqrt{\left[\mathbf{I}(\hat{\theta})^{-1} \right]_{ii}} \right)$$

Here, $\hat{\theta}_i$ is the estimate, I_{ii} is the inverse Fisher information matrix element, and $z_{\alpha/2}$ is the critical value for the desired confidence level. Therefore for:

Confidence Intervals (95.0 %):

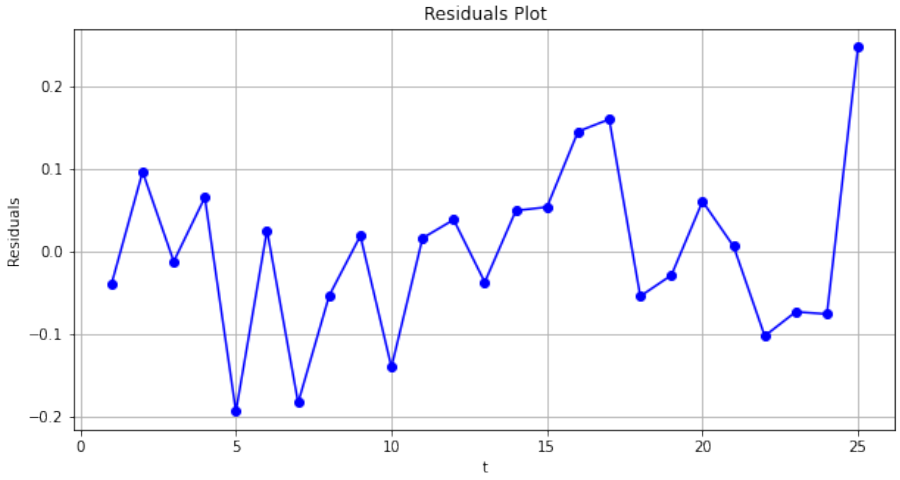
- α_1: (5.2124, 5.2962)
- α_2: (2.7877, 3.4429)
- β: (-0.7331, -0.6377)

```python
# Calculate the Fisher information matrix
def fisher_information(alpha1, alpha2, beta, t, residuals):
    n = len(t)
    var_residuals = np.var(residuals)
    d_alpha1 = np.ones(n) / var_residuals
    d_alpha2 = np.exp(2 * beta * t) / var_residuals
    d_beta = 2 * alpha2 * t * np.exp(2 * beta * t) / var_residuals
    fisher_matrix = np.array([
        [np.sum(d_alpha1), np.sum(d_alpha2), np.sum(d_beta)],
        [np.sum(d_alpha2), np.sum(d_alpha2 ** 2), np.sum(d_alpha2 * d_beta)],
        [np.sum(d_beta), np.sum(d_alpha2 * d_beta), np.sum(d_beta ** 2)]
    ])
    return fisher_matrix

fisher_matrix = fisher_information(alpha1_est, alpha2_est, beta_est, np.arange(1, 26), residuals)

# Calculate the standard errors
covariance_matrix = np.linalg.inv(fisher_matrix)
standard_errors = np.sqrt(np.diag(covariance_matrix))
# Define the confidence level (e.g., 95% confidence interval)
confidence_level = 0.95

# Calculate the critical t-value for the confidence interval
alpha = 1 - confidence_level
t_critical = t.ppf(1 - alpha / 2, len(y_observed) - 3)  # df = n - number of parameters

# Construct confidence intervals for each parameter
conf_int_alpha1 = (alpha1_est - t_critical * standard_errors[0], alpha1_est + t_critical * standard_errors[0])
conf_int_alpha2 = (alpha2_est - t_critical * standard_errors[1], alpha2_est + t_critical * standard_errors[1])
conf_int_beta = (beta_est - t_critical * standard_errors[2], beta_est + t_critical * standard_errors[2])
```

Part(6)

After obtaining parameter estimates, it is essential to analyze the residuals to assess the model fit. Residuals are the differences between the observed values and the values predicted by the model. A common practice is to create a plot of residuals against the predicted values or the variable t.

Residual Plot:



Part(7)

The Shapiro-Wilk test is a statistical test that evaluates whether a given sample comes from a normally distributed population. The test statistic is defined as:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where $x_{(i)}$ is the i-th ordered value, a_i are constants from the distribution of order statistics under normality, and $\bar{x}$ is the sample mean. The null hypothesis (H_0) is that the data follows a normal distribution.

The Shapiro-Wilk test was performed on the residuals, yielding the following results:

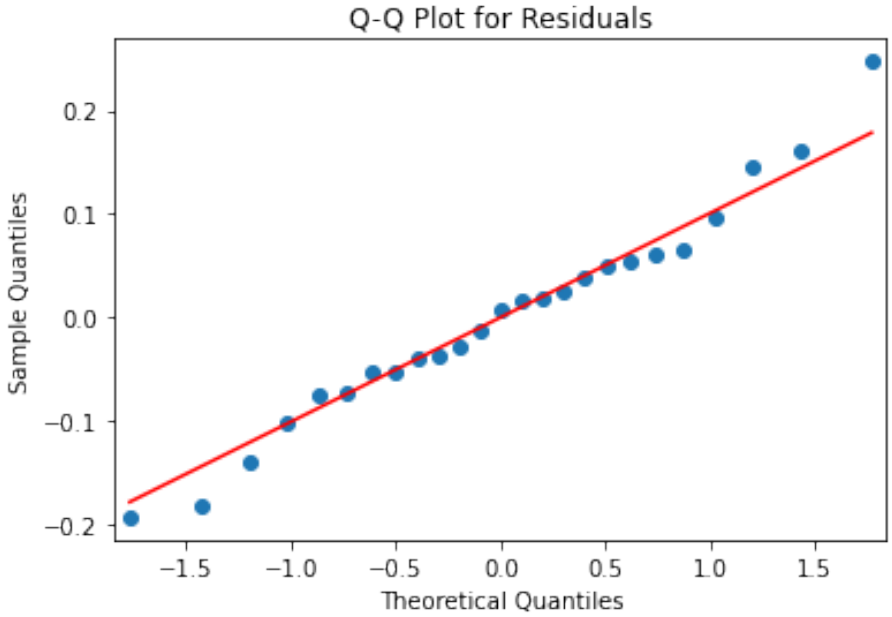
Statistic: 0.9823196530342102

P-value: 0.9268772006034851

Since the p-value (0.9269) is greater than the chosen significance level (usually 0.05), we fail to reject H_0. This suggests that the residuals appear to be normally distributed.

Quantile-Quantile (Q-Q) Plot

A Q-Q plot is a graphical tool used to assess the normality assumption. In the context of residuals, the Q-Q plot compares the quantiles of the observed residuals to the quantiles of a theoretical normal distribution.



The combination of statistical testing and graphical analysis provides a robust evaluation of the normality assumption is met by the residuals. The Shapiro-Wilk test was performed on the

residuals, yielding the following results:

$$\text{Statistic: } 0.9823196530342102$$
$$\text{P-value: } 0.9268772006034851$$

Since the p-value (0.9269) is greater than the chosen significance level (usually 0.05), we fail to reject H_0. This suggests that the residuals appear to be normally distributed.

Part (8)

Three-Dimensional Optimization for Least Squares Estimation

To obtain the least squares estimators of unknown parameters, a three-dimensional optimization approach is employed. The objective is to minimize the sum of squared residuals, given by:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{n} [y_i - f_i(\boldsymbol{\theta})]^2$$

Here, $\boldsymbol{\theta}$ represents the vector of unknown parameters, y_i is the observed value, and $f_i(\boldsymbol{\theta})$ is the model prediction for the i-th observation.

Optimization Method

The optimization is performed using the Nelder-Mead algorithm, a derivative-free optimization method available in R's 'optim' function. The Nelder-Mead method iteratively adjusts the parameters to minimize the objective function $Q(\boldsymbol{\theta})$. This method is suitable for optimizing functions without explicit derivatives.

Selection of Initial Values

The initial values can be found from part 2 where we minimised residual sum of squares as a function of β .The algorithm converges to the least squares estimators that provide the best fit of the model to the observed data.

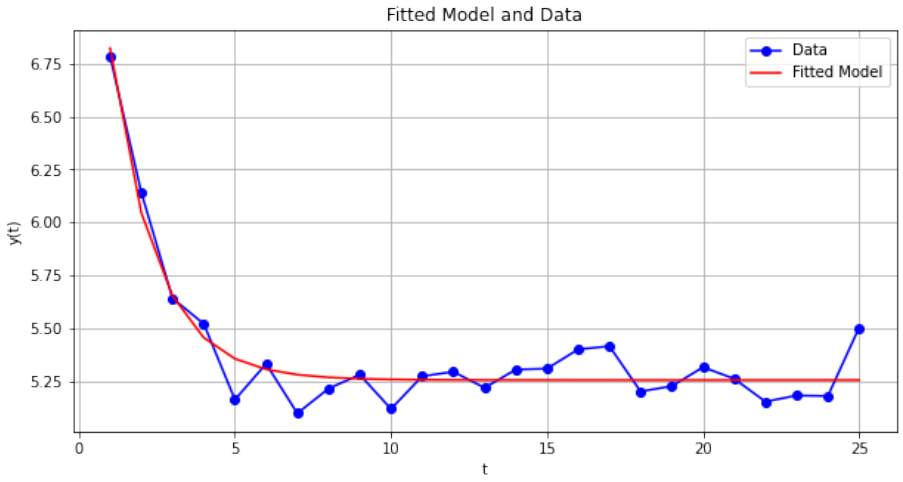
$$\text{Initial Value for } \alpha_1 : 1.0$$
$$\text{Initial Value for } \alpha_2 : 1.0$$
$$\text{Initial Value for } \beta : 0.72$$

After optimization, the final estimated parameters are found to be:

$$\text{Estimated } \alpha_1 : 5.254293777650497$$
$$\text{Estimated } \alpha_2 : 3.1153066828267457$$
$$\text{Estimated } \beta : -0.685408710728477$$

```python
# Define the model function
def model(params, t):
    alpha1, alpha2, beta = params
    return alpha1 + alpha2 * np.exp(beta * t)

# Define the residual function to be minimized
def residuals(params):
    y_pred = model(params, t)
    return np.sum((y_pred - y)**2)

# Initial guess for parameters
initial_guess = [1.0, 1.0, 0.72]

# Perform the optimization to estimate parameters
result = minimize(residuals, initial_guess, method='Nelder-Mead')
alpha1, alpha2, beta = result.x

# Parameter estimates
print("Estimated alpha1:", alpha1)
print("Estimated alpha2:", alpha2)
print("Estimated beta:", beta)
```

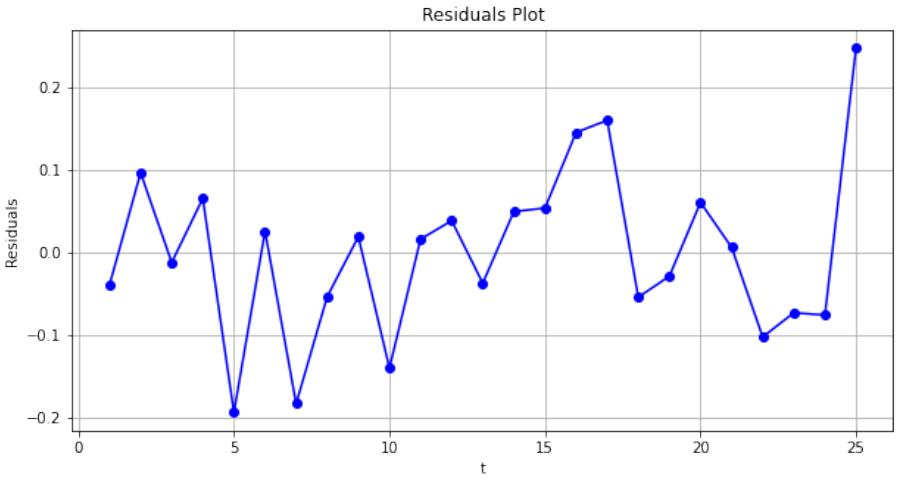
Figure 1: Python Code

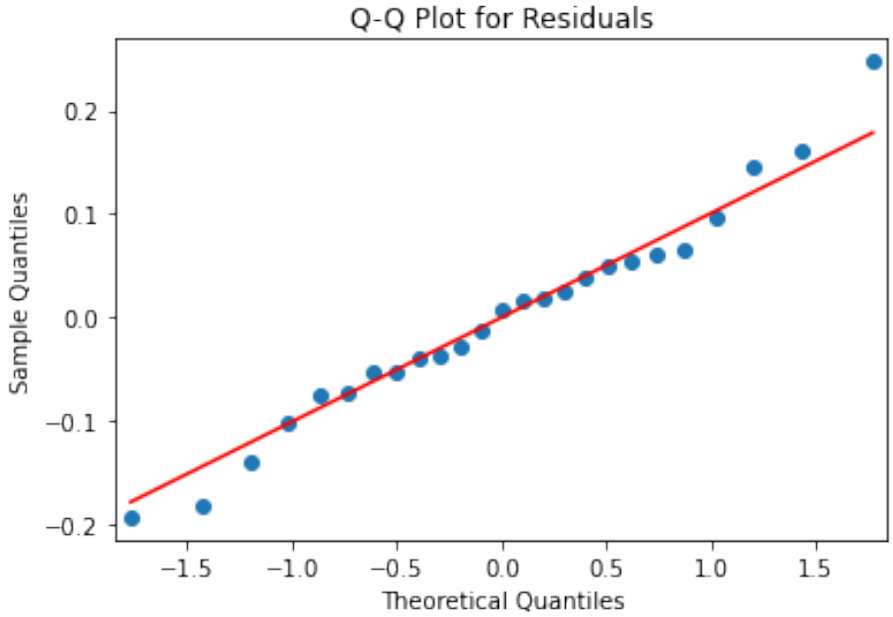


Estimated σ^2: 0.010202

For Confidence Intervals (95.0 %):

- α_1: (5.2124, 5.2962)
- α_2: (2.7877, 3.4428)
- β: (-0.7331, -0.6377)





The normal condition is also satisfied. All the values are pretty much the same indicating that our model is correct.

Part(9)

To fit the model $y(t) = \alpha_1 e^{\beta_1 t} + \alpha_2 e^{\beta_2 t} + \varepsilon(t)$ to the same data set, we continue with a similar optimization approach as before.

Model Representation

The extended model is expressed as a sum of two exponential terms and a random error term:

$$y(t) = \alpha_1 e^{\beta_1 t} + \alpha_2 e^{\beta_2 t} + \varepsilon(t)$$

Optimization Approach

The goal remains to minimize the sum of squared residuals, given by:

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{n} [y_i - f_i(\boldsymbol{\theta})]^2$$

Here, $f_i(\boldsymbol{\theta})$ is the model prediction for the i-th observation, which follows the extended model. Initial values will be dificult to estimate as their 4 parameters thus we had to use prony's equation which isn't taught. So let the intial values as [1.0, 0.1, 1.0, 0.2] for simplicity.

```
def model(params, t):
    α1, β1, α2, β2 = params
    return α1 * np.exp(β1 * t) + α2 * np.exp(β2 * t)
def objective(params, t, y):
    y_pred = model(params, t)
    residuals = y - y_pred
    return np.sum(residuals ** 2)
initial_params = [1.0, 0.1, 1.0, 0.2]
result = minimize(objective, initial_params, args=(t, y), method='Nelder-Mead')
α1_hat, β1_hat, α2_hat, β2_hat = result.x
print("Estimated Parameters:")
print("α1_hat =", α1_hat)
print("β1_hat =", β1_hat)
print("α2_hat =", α2_hat)
print("β2_hat =", β2_hat)

y_pred = model([α1_hat, β1_hat, α2_hat, β2_hat], t)
residuals = y - y_pred
ssr = np.sum(residuals ** 2)
print("Sum of Squared Residuals (SSR):", ssr)

plt.figure(figsize=(10, 5))
plt.plot(t, y, marker='o', linestyle='-', color='b', label='Data')
plt.plot(t, y_pred, linestyle='-', color='r', label='Fitted Model')
plt.xlabel('t')
plt.ylabel('y(t)')
plt.title('Fitted Model and Data')
plt.legend()
plt.grid(True)
plt.show()
```

Estimated Parameters for Extended Model

The estimated parameters and sum of squared residuals are as follows:

$$\text{Estimated } \alpha_1 : 5.156004314221248$$
$$\text{Estimated } \beta_1 : 0.0011832245506851551$$
$$\text{Estimated } \alpha_2 : 3.0493087032651123$$
$$\text{Estimated } \beta_2 : -0.6112223883059776$$
$$\text{Sum of Squared Residuals (SSR)} : 0.23257211019252966$$