Summer 2024

A Comparative Analysis of a Family of Advanced Iterative Optimization Methods in Nonlinear Regression

Tanmoy Kumar Debnath

A COMPARATIVE ANALYSIS OF A FAMILY OF ADVANCED ITERATIVE OPTIMIZATION METHODS IN NONLINEAR REGRESSION

by

TANMOY KUMAR DEBNATH

(Under the Direction of Divine Wanduku)

ABSTRACT

Classical statistical supervised learning optimization techniques like the Gauss-Newton Iterative Method (GNIM), Weighted Gauss-Newton Iterative Method (WGNIM), Reweighted Gauss-Newton Iterative Method (RGNIM), and Levenberg-Marquart (LM) algorithm extend the nonlinear least squares method. The WGNIM improves model fitting by controlling heteroscedasticity in the linear and nonlinear models. A comparative analysis of the GNIM, WGNIM, RGNIM, and LM methods for fitting nonlinear models is presented. A step-wise diagnosis for structural multicollinearity in the reweighted linearized model is investigated via the Variance Inflation Factor (VIF) to determine variance inflation in the sequence of estimators for the model parameters. Under restricted multicollinearity levels in simulated experiments, the RGNIM outperforms the GNIM with respect to precision, while the LM is most flexible for selecting the initial parameter estimate among all of the algorithms. Meanwhile, RGNIM and WGNIM have longer computational times.

INDEX WORDS: Nonlinear regression, Nonlinear least squares, Gauss-Newton method, Weighted least squares, Heteroscedasticity, Variance inflation factor, Statistical inference

2009 Mathematics Subject Classification: 62J02, 62J20, 65K10, 49M37

A COMPARATIVE ANALYSIS OF A FAMILY OF ADVANCED ITERATIVE OPTIMIZATION METHODS IN NONLINEAR REGRESSION

by

TANMOY KUMAR DEBNATH

B.S. in Mathematics, University of Dhaka, Bangladesh, 2016

M.S. in Applied Mathematics, University of Dhaka, Bangladesh, 2017

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN MATHEMATICS

A COMPARATIVE ANALYSIS OF A FAMILY OF ADVANCED ITERATIVE OPTIMIZATION METHODS IN NONLINEAR REGRESSION

by

TANMOY KUMAR DEBNATH

Major Professor:	Divine Wanduku
Committee:	Charles Champ
	Stephen Carden
	Andrew Sills

Electronic Version Approved:
July 2024

DEDICATION

To my beloved parents, all of my respected teachers, and my loving wife, whose unwavering love and support have been instrumental in helping me reach this point in my academic journey.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my thesis advisor, Dr. Divine Wanduku, for his invaluable guidance, assistance, and motivation throughout the course of this research and thesis. His mentorship not only improved my research skills but also instilled in me the values of patience and persistence. I am profoundly grateful for the opportunity to have Dr. Wanduku as my thesis supervisor.

I extend my gratitude to the exceptional instructors and staff at Georgia Southern University, whose combined knowledge and expertise have significantly enriched my academic journey. I am especially grateful to Dr. Hua Wang and Dr. Yi Hu for their steadfast guidance, as well as to Dr. Jiehua Zhu, Dr. Yan Wu, Dr. Goran Lesaja, Dr. Emil Iacob, and Dr. Scott Kersey for their unwavering support and substantial contributions to my educational experience. Their dedication to teaching is exemplary.

Furthermore, I wish to express my utmost gratitude to Dr. Charles Champ, Dr. Stephen Carden, and Dr. Andrew Sills for their generous willingness to serve as members of my thesis committee. Their profound understanding and specialized knowledge have greatly elevated the quality of my thesis.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In supervised learning, techniques for optimization are indispensable for statistical modeling where accurate model fitting is required. Notable strategies that effectively improve the nonlinear least squares approach are the Levenberg-Marquardt (LM) method, the Gauss-Newton Iterative Method (GNIM), the Weighted Gauss-Newton Iterative Method (WGNIM), and the Reweighted Gauss-Newton Iterative Method (RGNIM) [1]. When dealing with complicated data structures and model dynamics, these methods are vital for optimizing model fits through parameter value adjustment.

A classic approach, the GNIM method systematically adjusts a model's parameters to reduce the sum of squared differences between the observed and predicted values [3], [6], [10]. Heteroscedasticity, in which the residuals' variability is not constant across multiple levels of an explanatory variable, can be a challenge for this effective strategy [7]. Inefficient parameter estimations and incorrect statistical inferences might be caused by heteroscedasticity [8], [9].

In response to this difficulty, the WGNIM improves the reliability and robustness of the model fitting process by introducing a weighting mechanism that accounts for heteroscedasticity. The WGNIM improves model performance and produces more precise parameter estimates by using weights obtained from the observed data to scale the impact of each observation. That means the heteroscedasticity was effectively mitigated by the WGNIM, which produced more consistent variance and, consequently, more dependable confidence intervals. In comparison to the GNIM, the WGNIM needed fewer iterations to converge to an optimal solution, making it more efficient overall, even if it took more time per iteration [2], [3], [7].

Conversely, the adaptability of the LM approach is well known when choosing preliminary approximations of the parameters. It is a well-rounded method that successfully

explores complicated parameter spaces by integrating the best features of gradient descent and the Gauss-Newton method. Since the selection of initial values is pivotal for convergence in extremely nonlinear models, the LM technique becomes invaluable in such cases [10], [11], [12], [13].

Additionally, the Variance Inflation Factor (VIF) was employed to systematically identify structural multicollinearity in the reweighted linearized model. This analysis identified and controlled for variance inflation in the sequence of estimators, ensuring the stability and reliability of parameter estimates in simulated experiments with restricted multicollinearity levels. In contrast, the LM method exhibited greater adaptability when estimating parameters, whereas the RGNIM demonstrated superior performance over the GNIM in managing heteroscedasticity and enhancing model fit [11], [12], [14].

These techniques are used to examine a dataset of ultrasonic calibrations provided by the National Institute of Standards and Technology (NIST) in this research [15]. Ultrasonic calibration is the process of fine-tuning ultrasonic instruments so that they provide reliable measurement results. The dataset utilized for this research illustrates a real-world situation where the metal distance (predictor variable) affects the ultrasonic response (response variable). The initial analysis using the GNIM highlighted the presence of heteroscedasticity, which impaired the accuracy of the parameter estimates.

1.1 OUTLINE

This study is comprised of six chapters, each addressing a specific aspect of the research:

Chapter 1 will provide a detailed background on the work and offer a comprehensive overview of key concepts in statistical learning.

Chapter 2 will provide a comprehensive explanation of the multiple linear regression model, including its assumptions, ordinary least squares (OLS) regression, parameter

estimation, and their properties, as well as the method of maximum likelihood estimation of the parameters.

Chapter 3 will focus on multicollinearity, covering its different types, consequences, techniques for identification, and remedies. It will also include a complete derivation of generalized least squares estimation, followed by the derivation of weighted least squares estimation, and will discuss their assumptions and the selection of weights in detail.

Chapter 4 will introduce the concepts of nonlinear regression models, discussing their assumptions, differences from linear regression models, different types of nonlinear regression, and parameter estimation using the nonlinear least squares method. It will cover a complete derivation of the Gauss-Newton Iterative Method (GNIM) for nonlinear regression, with an example using the logistic growth model.

Chapter 5 will discuss the RGNIM and its application, using the logistic growth model as an example.

Chapter 6 will address heteroscedasticity and explore the application of GNIM and WGNIM to real-life data, specifically an ultrasonic calibration dataset. It includes the interpretation of maximum likelihood estimates for the regression coefficients. Finally, it will provide the findings of this study.

1.2 Some concepts in statistical learning

This section will provide a brief overview of key ideas in statistical learning and establish their relationship to the research methodologies utilized in this study.

1.2.1 What is statistical learning?

Data scientists can analyze and predict outcomes from datasets with the use of statistical learning. It is a branch of machine learning concerned with finding connections and patterns in data using statistical techniques. In other words, statistical learning is a collec-

tion of methods for making data-driven estimates of the relationship between variables. Its purpose is to help with data comprehension and prediction.

A common objective in statistical learning is to refine a model to reliably use previously unknown data to generate predictions or decisions. A training dataset of samples with known input variables and their corresponding output variables is used to understand these underlying patterns or correlations in the data. In general, the symbol X is used to represent the input variables, accompanied by a subscript to differentiate them. These inputs are referred to by several names: **predictors, features, independent variables, or simply variables**. Most commonly, the output variable is represented by the letter Y. It is also referred to as the **response or dependent variable** [13].

Consider a quantitative response variable denoted by Y and p distinct predictors, denoted by $X_1, X_2, ..., X_p$. We make the assumption that a relationship exists between Y and $X = (X_1, X_2, ..., X_p)$, which can be expressed in the most general sense as

$$Y = f(X) + \epsilon. \tag{1.1}$$

In this context, f represents an unidentified fixed function of $X_1, X_2, ..., X_p$, and ϵ denotes a random error term with a mean of zero that is independent of X. The formula denotes f as the systematic information provided by X regarding Y.

Fundamentally, statistical learning represents a collection of methodologies utilized to approximate f.

Principal purposes for statistical modeling are as follows:

1. Prediction

2. Inference

Prediction

There are numerous cases, despite the availability of a set of inputs X, the output Y is not always easy to produce. Given this configuration, where the error term averages to zero, we can use the following formula to forecast Y:

$$\hat{Y} = \hat{f}(X), \tag{1.2}$$

where $\hat{f}$ is our estimate of f and $\hat{Y}$ denotes the final prediction for Y [13].

The precision of $\hat{Y}$ as a prediction for Y relies on two factors: the reducible error and the irreducible error.

Reducible Error

In statistical modeling, the term **"reducible error"** describes the portion of the total error in prediction that may be mitigated by model improvement. The situation occurs due to the model's imperfect representation of the basic relationship that exists between the predictors and the response variable. Generally, the estimate $\hat{f}$ will not be an exact representation of f, leading to some degree of error. This error is called a reducible error [13]. This error can be minimized by enhancing the accuracy of $\hat{f}$ by utilizing the most suitable statistical learning approach to estimate f.

For example, it is possible to reduce the inaccuracy produced by oversimplification in a medical study that attempts to predict the risk of heart disease based on cholesterol levels by limiting the model to only include total cholesterol and ignoring other important factors such as age, blood pressure, and family history. Including these extra components in the model can help reduce this error.

Irreducible Error

Irreducible error in statistical modeling is the part of the total error that cannot be reduced by improving the model. It is caused by inherent randomness, variability, or unpredictability in the data or the underlying process being modeled. That means, even with a precise estimate for f as $\hat{Y} = \hat{f}(X)$, the prediction would still contain some inaccuracy. Y is a function of ϵ, which cannot be predicted given X. Thus, the variability linked to ϵ impacts the precision of our predictions. This is referred to as the irreducible error, as it cannot be minimized regardless of how well estimate f is due to the inaccuracy caused by ϵ [13].

An explicit illustration of irreducible error can be observed when trying to predict the precise arrival time of a vehicle at a designated stop. There are elements that cannot be precisely anticipated or controlled, including traffic congestion, road conditions, and unanticipated delays, despite the utilization of the most sophisticated prediction models. The unpredictability of these factors and the impossibility of eliminating them through model enhancements contribute to the irreducible error in the arrival time prediction.

The prediction $\hat{Y} = \hat{f}(X)$ is obtained by utilizing a set of predictors X and a given estimate $\hat{f}$. Consider that $\hat{f}$ and X are both fixed, with random error term ϵ. Then mathematically,

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(x) + \epsilon - \hat{f}(x)]^2 \\ &= E[f(x) - \hat{f}(x) + \epsilon]^2 \\ &= E[(f(x) - \hat{f}(x))^2 + 2(f(x) - \hat{f}(x))\epsilon + \epsilon^2] \\ &= (f(x) - \hat{f}(x))^2 + 2(f(x) - \hat{f}(x))E(\epsilon) + E(\epsilon^2) \end{aligned}$$

[**Note:** Since $E(Z + c) = E(Z) + c$, $E(cZ) = cE(Z)$ and $E(c) = c$, where Z is random

and c is a constant.]

$$= (f(x) - \hat{f}(x))^2 + 2(f(x) - \hat{f}(x))(0) + E(\epsilon^2)$$
$$= (f(x) - \hat{f}(x))^2 + E(\epsilon^2)$$
$$= (f(x) - \hat{f}(x))^2 + \mathbf{Var}(\epsilon)$$

[**Note:** Since $\mathbf{Var}(\epsilon) = E(\epsilon^2) - E(\epsilon)^2 = E(\epsilon^2) - 0 = E(\epsilon^2)$]

Therefore,

$$E(Y - \hat{Y})^2 = (f(x) - \hat{f}(x))^2 + \mathbf{Var}(\epsilon),$$

where $E(Y - \hat{Y})^2$ represents the **average** or **expected value** of the squared difference between the predicted and actual value of Y, the term $(f(x) - \hat{f}(x))^2$ represents **reducible error** and $\mathbf{Var}(\epsilon)$ represents the variance associated with the random error term, which is **irreducible error** [13].

Remark 1.1. *Although the independent variables or predictors in linear regression and nonlinear regression are usually considered to be fixed, the predictors cannot be fixed in some cases. For instance, in the case of time-related observations, the predictors will change over time.*

Inference

In some circumstances, it is crucial to comprehend the manner in which a change in X influences variable Y. Although estimating f is the objective in this circumstance, making predictions for Y is not necessarily the aim. Understanding the relationship between X and Y is the primary focus at this time, and the precise form of f is the subject of concern in this instance. It is referred to as an inference [13].

For instance, instead of trying to predict an individual's blood pressure using the medication dose, research investigating the effects of a new medicine (X) on blood pressure (Y) may aim to understand how changes in the dosage of the drug influence blood

pressure. The primary focus here is on the specific shape of the function f, which represents the link between medicine dose and blood pressure.

Let us assume a simple linear regression equation:

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

where Y represents the dependent variable (e.g. blood pressure), X represents the independent variable (e.g. medicine dosage), β_0 represents the intercept term, in the above example it expressing the expected blood pressure when the medicine dosage is 0, β_1 represents the slope coefficient, in the above example it representing the change in blood pressure for a one-unit change in medicine dosage, ϵ represents the error term, which is the difference between the observed blood pressure and the predicted blood pressure.

The purpose of inference in this equation is to approximate the values of β_0 and β_1 in order to comprehend the impact of drug dosage variations on blood pressure. As an example, if β_1 is expected to be -3.5, it means that blood pressure typically drops by 3.5 units for every extra unit of medicine dosage. To ascertain the efficacy and safety of the medicine, the inference is necessary to comprehend the connection between the dose and blood pressure.

1.2.2 Supervised and Unsupervised learning

There are two main types of statistical learning problems [13]:

1. Supervised Learning
2. Unsupervised Learning

The usage of labeled datasets is the key distinctive feature between the supervised and unsupervised learning. A supervised learning algorithm relies on labelled input and output data, whereas an algorithm for unsupervised learning does not does not utilized the labelled input and output data.

In addition to, there are more types of learning, such as **"semi-supervised"** learning that employs both labelled and unlabeled data, and **"reinforcement"** learning that is sequential decision-making to maximize long-term reward [13].

Supervised Learning

Supervised learning is a problem type in machine learning. Supervised learning involves using labeled datasets to train algorithms to forecast the output. In supervised learning a corresponding response measurement $y_i, \forall i = 1, 2, ..., n$ is connected with each observation of the predictor measurement(s) $x_i, \forall i = 1, 2, ..., n$. The goal of supervised learning is to establish a connection between the response and the predictors, either for improved understanding of the relationship between the response and predictor (inference) or for more accurate prediction of the response for future observations (prediction) [13].

Unsupervised Learning

A second type of machine learning problem is unsupervised learning. In unsupervised learning, there are no response values available, that means, there is no predefined labels for the input data; we only have a set of predictor values $x_i, \forall i = 1, 2, ..., n$. Detecting concealed patterns or intrinsic structures within the data is the salient purpose of unsupervised learning. In unsupervised learning, clustering is a commonly used technique in which the algorithm clusters together data elements that are similar [13]. Dimensionality reduction is another problem where the algorithm reduces the number of input variables by keeping important information. Popular techniques for unsupervised learning are K-means clustering, hierarchical clustering, and principal component analysis (PCA).

1.2.3 REGRESSION AND CLASSIFICATION PROBLEMS

There are two main types of variables: **quantitative** and **qualitative** (often called **categorical**). Numerical values are assigned to quantitative variables. As for example the market price of a house, the stock price, and a person's age, height, and income [13].

Qualitative variables, on the other hand, may only take on values that belong to one of a set of K distinct groups [13]. As for example, the brand of a product (A, B, or C), marital status of a person (married or unmarried), and so on.

Based on input data, regression algorithms forecast continuous values. In regression problems, input and output variables are used to estimate a function of a model. The regression model is appropriate for quantities like salary, height, or weight, age, value of property. Depending on the different problems and situations, data scientists and the engineers of machine learning utilize different regressions in statistical issues. There are different types of regression algorithm like simple linear regression, multiple linear regression, polynomial regression, nonlinear regression and so on.

Now the question is why nonlinear regression is a regression problem? Like linear regression, nonlinear regression predicts a continuous outcome variable from one or more predictor factors. The main distinction is that nonlinear regression doesn't assume a linear connection between predictor and response variables. It always follows different types of nonlinear pattern. Nonlinear regression attempts to determine the parameters of a selected nonlinear model that most precisely fits the data. This is usually achieved by reducing the gap between the actual values and the values predicted by the nonlinear model, employing techniques like the Gauss-Newton algorithm or the Levenberg-Marquardt algorithm and so on.

On the other hand, classification is a kind of predictive modeling that uses input variables to estimate a mapping function that identifies discrete labels or categories as output variables. Predicting the category or label of the input variables is the vital role of

the mapping function in classification algorithms. Regardless of whether the variables in a classification method are discrete or real-valued, it is necessary for the instances to be categorized into at least two classes. There are different types of classification algorithms like random forest classification, decision tree classifications, K-nearest neighbour classification etc. [13].

1.2.4 Assessing accuracy and precision in the regression model

Evaluating the Accuracy and Precision of the Fit

To assess the efficacy of a technique on a certain collection of dataset, it is necessary to estimate the discrepancy between the actual and anticipated response. That means, the residuals are the discrepancies between the model's predicted values and the observed data values that occur when attempting to fit a regression model to predict a continuous response variable and then uses that model to forecast the values of some data. In regression analysis, Mean Squared Error (MSE) is the most widely used metric [13]. MSE is measured as the average of the residuals of a model which is given by

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2. \tag{1.3}$$

In the above equation (1.3), the forecast value $\hat{f}$ that provides for the i^{th} observation is denoted as $\hat{f}(x_i)$.

Remark 1.2. *The Mean Squared Error (MSE) will be minimized when the expected responses closely match the actual responses, and will be maximized when there is a significant difference between the predicted and true responses for some observations.*

Training Dataset

A machine learning model is trained using a subset of a dataset known as the training dataset. A collection of input-output pairs is utilized to instruct the model on the relationship between inputs and outputs. The model gains knowledge from this dataset by

modifying its internal parameters in response to the input data and the corresponding accurate output.

Example 1.2.1. *Consider a database that contains house-related data, such as dimensions, number of bedrooms, geographical location, and their respective costs. This is one possible structure for a machine learning dataset that might be used to forecast home prices:*

Table 1.1: Training Dataset

Size (sq ft)	Bedrooms	Location	Price (Dollar)
1500	3	Suburban	250,000
2000	4	Urban	300,000
⋮	⋮	⋮	⋮

By utilizing the provided training dataset, the model shall assimilate knowledge regarding the association between the output (price) and the input attributes (size, bedrooms, location).

Test Dataset

To evaluate the efficacy of a machine learning model that has been trained, a distinct subset of the dataset is designated as the "test dataset". It comprises pairings of inputs and outputs that were not seen by the model during the training process. A performance evaluation of the model is conducted by comparing the accuracy of its predictions to the observed outputs, which are generated using the test dataset.

Example 1.2.2. *To further elaborate on the example of house price prediction, consider the following as a possible test dataset:*

Table 1.2: Test Dataset

Size (sq ft)	Bedrooms	Location	Price (Dollar)
1500	3	Suburban	?
1200	4	Rural	?
⋮	⋮	⋮	⋮

The price predictions for these houses will be generated by the model utilizing the learned parameters. The model's performance will be assessed according to the degree of correspondence between these forecasts and the real prices.

Training MSE

This is the mean squared error (MSE) obtained on the identical dataset that was utilized for training the model. The metric calculates the mean squared deviation between the observed values and the predicted values generated by the model using the training dataset. A low training mean squared error (MSE) suggests that the model is effectively capturing the patterns in the training data [13].

Test MSE

The Mean Squared Error (MSE) is computed on a distinct dataset, referred to as the test dataset, which the model has not been shown during the training process. It assesses the extent to which the model can effectively apply its learned knowledge to unfamiliar data. A low mean squared error (MSE) suggests that the model has the capability to generate precise predictions on unfamiliar data.

In general, it is desirable to have a low MSE for both the training and test phases. **"Overfitting"** may occur when the training mean square error (MSE) is significantly smaller than the test MSE [13]. This occurs when the model becomes overly intricate with the training data, thereby capturing extraneous noise rather than the true pattern. The presence of

an underfitting condition, wherein the model fails to represent the underlying pattern in the data, may be suggested if the test MSE is significantly greater than the training MSE.

Bias-Variance Trade-Off

What is Bias?

Bias refers to the model's incapacity to accurately predict values, resulting in differences or errors between the average prediction of the model and the actual values. The disparities between the actual or anticipated values and the projected values are referred to as errors, specifically bias errors or errors resulting from bias [13].

What is Variance?

A data set's variance indicates how far individual values deviate from the mean. A predictive model's variance in machine learning is the degree to which its performance deviates from the mean when trained on various data subsets. In particular, the model's variance is its sensitivity to a different subset of the training dataset, or its ability to adapt to the new subset [13].

The expected test MSE of a technique is rely on its variance and bias. Consider that x_0 and y_0 are fixed and $E[(y_0 - \hat{f}(x_0))^2]$ is the average test MSE, where $\hat{f}$ is estimated by using different training datasets. Then

$$\begin{aligned} E[(y_0 - \hat{f}(x_0))^2] &= E[(y - \hat{f})^2] \\ &= E[(y - f + f - \hat{f})^2)] \\ &= E[(y - f)^2 + (f - \hat{f})^2 - 2(y - f)(f - \hat{f})] \\ &= E[(y - f)^2] + E[(f - \hat{f})^2] - 2E[(y - f)(f - \hat{f})] \\ &= E[\epsilon^2] + E[(f - \hat{f})^2] - 2E[(y - f)(f - \hat{f})] \\ &= \mathbf{Var}(\epsilon) + E[(\hat{f} - f)^2] - 2E[(y - f)(f - \hat{f})]. \end{aligned} \tag{1.4}$$

$$\begin{aligned} E[(\hat{f} - f)^2] &= E[(\hat{f} - E(\hat{f}) + E(\hat{f}) - f)^2] \\ &= E[(\hat{f} - E(\hat{f}))^2 + (E(\hat{f}) - f)^2 + 2(\hat{f} - E(\hat{f}))(E(\hat{f}) - f)] \\ &= E[(\hat{f} - E(\hat{f}))^2] + E[(E(\hat{f}) - f)^2] + 2E[(\hat{f} - E(\hat{f}))(E(\hat{f}) - f)] \\ &= \mathbf{Var}(\hat{f}) + [\mathbf{Bias}(\hat{f})] + 2E[(\hat{f} - E(\hat{f}))(E(\hat{f}) - f)]. \end{aligned} \tag{1.5}$$

$$\begin{aligned} 2E[(\hat{f} - E(\hat{f}))(E(\hat{f}) - f)] &= 2E[\hat{f}E[\hat{f}] - \hat{f}f - E[\hat{f}]E[\hat{f}] + E[\hat{f}]f] \\ &= E[\hat{f}]E[\hat{f}] - fE[\hat{f}] - E[\hat{f}]E[\hat{f}] + fE[\hat{f}] \\ &= 0 \end{aligned} \tag{1.6}$$

using (1.6), in equation (1.5),

$$E[(\hat{f} - f)^2] = \mathbf{Var}(\hat{f}) + [\mathbf{Bias}(\hat{f})] \tag{1.7}$$

$$\begin{aligned} E[(y - f)(f - \hat{f})] &= E[yf - y\hat{f} - f^2 + f\hat{f}] \\ &= f^2 - f^2 - E[y\hat{f}] + fE[\hat{f}] \\ &= -E[(y + \epsilon)\hat{f}] + fE[\hat{f}] \\ &= -E[f\hat{f}] - E[\epsilon\hat{f}] + fE[\hat{f}] \\ &= -fE[\hat{f}] - E[\epsilon\hat{f}] + fE[\hat{f}] \\ &= 0 \end{aligned} \tag{1.8}$$

using (1.7) and (1.8) in equation (1.4),

$$E[(y_0 - \hat{f}(x_0))^2] = \mathbf{Var}(\hat{f}(x_0)) + [\mathbf{Bias}(\hat{f}(x_0))] + \mathbf{Var}(\epsilon). \tag{1.9}$$

From equation (1.9) observe that the expected test MSE for a given value of x_0 can be decomposed into three fundamental quantities [13], [17]:

- The squared bias of the predicted functional form of $\hat{f}(x_0)$.

- The variance of the predicted functional form of $\hat{f}(x_0)$.
- The variance of the error terms ϵ.

Thus, in order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias.

Low Bias: Low bias value means fewer assumptions are taken to build the target function. In this case, the model will closely match the training dataset [13].

High Bias: High bias value means more assumptions are taken to build the target function. In this case, the model will not match the training dataset closely [13].

Variance refers to the amount by which the predicted functional form of $\hat{f}$ would change if we estimated it using a different training data set. Since the training data are used to fit the statistical learning method, different training data sets will result in a different $\hat{f}$.

In general, more flexible statistical methods have higher variance. As a general rule, as we use more flexible methods, the variance will increase and the bias will decrease. The relative rate of change of these two quantities determines whether the test MSE increases or decreases.

As we increase the flexibility of a class of methods, the bias tends to initially decrease faster than the variance increases. Consequently, the expected test MSE declines. However, at some point increasing flexibility has little impact on the bias but starts to significantly increase the variance. When this happens the test MSE increases.

The relationship between bias, variance, and test set MSE outlined above is referred to as the bias-variance trade-off [13].

Coefficient of Determination

The proportion of the variance in the dependent variable that can be predicted from the independent variables in a regression model is represented by the statistical measure known as the coefficient of determination, which is denoted as R^2. In simple terms, it

signifies the extent to which the variability of the dependent variable can be explained by the independent variables.

It is known that, the model provides an explanation for some variability but does not do so for all. The overall variability is given by the sum of these two factors.

$$\sum(y_i - \bar{y})^2 = \sum(\hat{y}_i - \bar{y})^2 + \sum(y_i - \hat{y}_i)^2. \tag{1.10}$$

Equation (1.10) can be written as

$$SST = SSR + SSE. \tag{1.11}$$

In the above equation (1.10) the left hand side term $\sum(y_i - \bar{y})^2$ is called total sum of squared which represents the total variation, in the right hand side the first term $\sum(\hat{y}_i - \bar{y})^2$ is called residual sum of squared which represents explained variation and the second term $\sum(y_i - \hat{y}_i)^2$ is called error sum of squared which represents unexplained variation [12].

SST is a measure of the uncertainty in predicting y when x is not considered. Similarly, SSE measures the variation in y_i when a regression model utilizing the predictor variable x is employed. A natural measure of the effect of x in reducing the variation in y, i.e., in reducing the uncertainty in predicting y, is to express the reduction in variation $(SST - SSE = SSR)$ as a proportion of the total variation [12], [14]:

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}, \tag{1.12}$$

where $\hat{y}_i$ is the predicted value of the dependent variable for observation, $\bar{y}$ is the mean of the observed values of the dependent variable, y_i is the observed value of the dependent variable for observation, n is the number of observations.

The measure R^2 is called the coefficient of determination, which is the statistical measure for evaluating the goodness of fit a regression model. Since $0 \leq SSE \leq SST$, then the range of the values of R^2 lies between 0 and 1, i.e.,

$$0 \leq R^2 \leq 1$$

- $R^2 = 0$ indicates that the independent variables do not explain any of the variability of the dependent variable.
- $R^2 = 1$ indicates that the independent variables explain all the variability of the dependent variable [12], [14].

Adjusted Coefficient of Determination

In a regression model, the adjusted coefficient of determination, represented as R^2_{adj}, is a revised form of the original coefficient of determination, R, that adjusts for the number of predictors. Its primary function is to offer a more precise assessment of the regression model's goodness of fit, particularly in cases where contrasting models have varying numbers of predictors.

Although the value of R tends to increase with the number of predictors added to a model, even if those predictors are irrelevant, the inclusion of extraneous predictors is penalized by R^2_{adj}. This feature serves to prevent overfitting and offers a more accurate evaluation of the model's performance.

The formula for calculating R^2_{adj} is:

$$\begin{aligned} R^2_{adj} &= 1 - \frac{\frac{SSE}{n-p}}{\frac{SST}{n-1}} \\ &= 1 - \frac{SSE}{SST}.\frac{(n-1)}{(n-p)} \\ &= 1 - \frac{(1-R^2)(n-1)}{n-p}, \end{aligned} \tag{1.13}$$

where n is the number of observations, p is the number of predictors in the model.

When adding additional predictors does not enhance the performance of the model, R^2_{adj} will consistently be equal to or less than R^2. For comparing the goodness of fit of different models, it is generally favored over R^2 because it provides a more conservative estimate of the proportion of variance explained by the model [12], [14].

CHAPTER 2

THE MULTIPLE LINEAR REGRESSION MODEL

2.1 MULTIPLE REGRESSION MODELS

We consider the linear regression model with a single predictor (regressor) variable. The model is stated as

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{2.1}$$

It is commonly referred to as the simple linear regression model because only one predictor variable is involved. The intercept β_0 and the slope β_1 are unknown constants and ϵ is a random error component. The errors are assumed to have mean zero and unknown variance σ^2. Additionally we usually assume that the errors are uncorrelated. This means that the value of one error does not depend on the value of any other error.

Consider, n pairs of dataset, say $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$. Then the above model (2.1) can be stated as

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \forall i = 1, 2, ..., n. \tag{2.2}$$

Equation (2.1) called as a **population regression model** while (2.2) is a **sample regression model** [12], written in terms of the n pairs of data $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, where y_i is the value of the response variable in the i^{th} observation, β_0 and β_1 are parameters, x_i is a known constant (the value of the predictor variable in the i^{th} observation), ϵ_i is a random error term with mean $E(\epsilon_i) = 0$ and variance $\mathbf{Var}(\epsilon_i) = \sigma^2$.

The extension of the linear regression model is the multiple linear regression model. Analysis of the association between a dependent variable and two or more independent variables is accomplished through the use of a statistical technique multiple regression model. The simplified form linear regression is essentially extended to situations involving multiple predictors.

In the multiple regression model, the response y is associated to k regressors or predictor variables. The model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon \tag{2.3}$$

is called a multiple linear regression model with k regressors. The sample regression model corresponding to equation (2.3) can be written as

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \epsilon_i \\ &= \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + \epsilon_i, \forall i = 1, 2, \cdots n, \end{aligned} \tag{2.4}$$

where error terms ϵ_i identical, independent and normally distributed with mean 0 and variance σ^2, which can be written as $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$. In vector form equation (2.4) can be written as,

$$\vec{y} = X\vec{\beta} + \vec{\epsilon}. \tag{2.5}$$

In matrix terms, it can be defined by the following matrices

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}, \vec{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}, \vec{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix},$$

where $\vec{y}$ is an $n \times 1$ vector of responses, X is an $n \times p$ matrix of the levels of the regressor variables (constant), $\vec{\beta}$ is a vector of parameters or the regression coefficients, and $\vec{\epsilon}$ is a vector of independent normal random errors.

The expectation of random error is $E(\vec{\epsilon}) = 0$ and variance-covariance matrix is

$$\mathbf{Var}(\epsilon) = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 I.$$

Consequently, the expectation of the random variable y is

$$E(y) = X\beta$$

and the variance-covariance matrix of y is

$$\mathbf{Var}(y) = \sigma^2 I.$$

2.1.1 ASSUMPTIONS

It is important to check a number of assumptions before applying a multiple regression. The assumptions are following [18]:

1. The values of the predictors, $x_{i1}, x_{i2}, \cdots, x_{ik}$, $\forall i = 1, 2, \cdots, n$, can be taken as constants; they are not random variables.

2. The expected value of ϵ is, $E(\epsilon_i) = 0, \forall i = 1, 2, \cdots, n$.

3. The errors, ϵ_i, at each set of values of the predictors, $x_{i1}, x_{i2}, \cdots, x_{ik}$, are normally distributed.

4. $\mathbf{Var}(\epsilon_i) = \sigma^2, \forall i = 1, 2, \cdots, n$, is constant. This implies that the variances $\mathbf{Var}(\vec{y}) = \sigma^2$ are all the same. All observations have the same precision.

5. The different random errors ϵ_i and ϵ_j, and their corresponding different responses y_i and y_j are independent. This implies that $\mathbf{Cov}(\epsilon_i, \epsilon_j) = 0$, for $i \neq j$.

2.2 ESTIMATION OF PARAMETERS IN THE MULTIPLE LINEAR REGRESSION MODEL

This section will describe the OLS method for estimating parameters in the multiple linear regression model.

2.2.1 ORDINARY LEAST SQUARES (OLS) REGRESSION

The method of "least squares" is a versatile mathematical technique employed to determine the optimal curve that best fits a given set of data points. It reduces the sum of the squared differences between the values that were seen and those that were projected. This approach is applicable to many kinds of equations and models.

The terms "ordinary least squares" (OLS) and "least squares" (LS) are often used interchangeably, but "ordinary least squares" (OLS) is a specific type of least squares method that is commonly used in the context of linear regression. A technique for estimating the unknown parameters in a linear regression model in statistics is called ordinary least squares (OLS) or linear least squares. Ordinary Least Squares (OLS) specifically refers to the method of linear regression that minimizes the sum of the squared differences between the responses predicted by the linear approximation and the observed responses in the dataset.

Ordinary Least Squared Estimation of the regression coefficient

Regression parameters of equation (2.3) can be estimated by using the method of least squares. Assume that there are more than k observations; for each iteration of the regressor x_j, let y_i represent the observed response and x_{ij} stand for the i^{th} observation. Consider that the errors are uncorrelated and that the model's error term ϵ has $E(\epsilon) = 0$ and $\mathbf{Var}(\epsilon) = \sigma^2$.

For given y and X, the object is to find out the vector of least-squared estimators, $\hat{\beta}$, that minimizes the sum of squared of ϵ, i.e,

$$
\begin{aligned}
S(\vec{\beta}) = \sum_{i=1}^{n} \epsilon_i^2 = (\vec{\epsilon})^{'}\vec{\epsilon} &= (\vec{y} - X\vec{\beta})^{'}(\vec{y} - X\vec{\beta}) \\
&= ((\vec{y})^{'} - (X\vec{\beta})^{'})(\vec{y} - X\vec{\beta}) \\
&= ((\vec{y})^{'} - (\vec{\beta})^{'}X^{'})(\vec{y} - X\vec{\beta}) \\
&= (\vec{y})^{'}\vec{y} - (\vec{y})^{'}X\vec{\beta} - (\vec{\beta})^{'}X^{'}\vec{y} + (\vec{\beta})^{'}X^{'}X\vec{y} \\
&= (\vec{y})^{'}\vec{y} - 2(\vec{\beta})^{'}X^{'}\vec{y} + (\vec{\beta})^{'}X^{'}X\vec{y}.
\end{aligned}
$$

Remark 2.1. *Here $\vec{\beta}$ is a $p \times 1$ vector, $(\vec{\beta})^{'}$ is a $1 \times p$ vector, X is a $n \times p$ matrix, $X^{'}$ is a $p \times n$ matrix, $\vec{y}$ is a $n \times 1$ vector. So the dimension of the matrix $(\vec{\beta})^{'}X^{'}\vec{y}$ is 1×1, which is a scalar, and its transpose $((\vec{\beta})^{'}X^{'}\vec{y})^{'} = (\vec{y})^{'}X\vec{\beta}$ has the same dimension, that is, the same scalar. $S(\vec{\beta})$ is a real valued and differentiable function.*

Now after differentiating on both sides of $S(\vec{\beta})$ with respect to β yields

$$
\begin{aligned}
\frac{\partial S}{\partial \vec{\beta}} &= -2X^{'}\vec{y} + 2X^{'}X\vec{\beta}, \\
\frac{\partial^2 S}{\partial (\vec{\beta})^2} &= 2X^{'}X,
\end{aligned}
$$

where $\frac{\partial^2 S}{\partial (\vec{\beta})^2}$ is non-negative definite. Then the least squared estimator must satisfy

$$
\left.\frac{\partial S}{\partial \vec{\beta}}\right|_{\hat{\vec{\beta}}} = -2X^{'}\vec{y} + 2X^{'}X\hat{\vec{\beta}} = 0,
$$

which simplifies to

$$
X^{'}X\hat{\vec{\beta}} = X^{'}\vec{y}. \tag{2.6}
$$

Equations (2.6) are called the least squares normal equations. If the predictors are linearly independent, that is, if columns of the X matrix can not be expressed as a linear combination of the other columns, which mathematically can be written as rank$(X) = k$(full rank), then $X^{'}X$ is a positive definite. So, there exist a inverse matrix of $X^{'}X$.

To figure out the least-squares estimator of $\vec{\beta}$, solve the normal equation (2.6). Now, multiplying on both sides of equation (2.6) by $(X^{'}X)^{-1}$ yields

$$\begin{aligned}(X^{'}X)^{-1}X^{'}X\hat{\vec{\beta}} &= (X^{'}X)^{-1}X^{'}\vec{y}\\ \text{or, } I\hat{\vec{\beta}} &= (X^{'}X)^{-1}X^{'}\vec{y}\\ \text{or, } \hat{\vec{\beta}} &= (X^{'}X)^{-1}X^{'}\vec{y},\end{aligned}$$

which is the required ordinary least squares estimator (OLSE) of $\vec{\beta}$. Since $\frac{\partial^2 S}{\partial(\vec{\beta})^2}$ is non-negative definite, so $\hat{\vec{\beta}}$ minimize $S(\vec{\beta})$.

Fitted values and Residuals

Fitted Values:

The fitted regression model is as follows

$$\hat{y} = X\hat{\vec{\beta}}, \tag{2.7}$$

where $\hat{\vec{\beta}}$ is the estimator of $\vec{\beta}$. Then

$$\begin{aligned}\hat{y} &= X\hat{\vec{\beta}}\\ &= X(X^{'}X)^{-1}X^{'}\vec{y}\\ &= H\vec{y},\end{aligned}$$

where $H = X(X^{'}X)^{-1}X^{'}$ is $n \times n$ matrix and which is called "Hat matrix". It transforms or maps the vector of observed values into a vector of fitted values [12], [19].

Properties of H:

- H is symmetric matrix.

- H is idempotent matrix, i.e, $HH = H$.

$$\begin{aligned} HH &= (X(X^{'}X)^{-1}X^{'})(X(X^{'}X)^{-1}X^{'}) \\ &= X(X^{'}X)^{-1}(X^{'}X)(X^{'}X)^{-1}X^{'} \\ &= H. \end{aligned}$$

Residuals:

It is known that the difference between the observed and fitted values is called residual, which mathematically can be written as [20]

$$\begin{aligned} \vec{e} &= \vec{y} - \hat{\vec{y}} \\ &= \vec{y} - X\hat{\vec{\beta}} \\ &= \vec{y} - X(X^{'}X)^{-1}X^{'}\vec{y} \\ &= \vec{y} - H\vec{y} \\ &= (I - H)\vec{y}. \end{aligned}$$

Here,

- $(I - H)$ is a symmetric matrix.
- $(I - H)$ is an Idempotent matrix, i.e, $(I - H)(I - H) = (I - H)$.

$$\begin{aligned} (I - H)(I - H) &= I - IH - HI + HH \\ &= I - 2H + HH \\ &= I - 2H + H \,(\because HH = H) \\ &= I - H. \end{aligned}$$

Properties of Least-Squares Estimators

The properties of least-squares estimators $\hat{\beta}$ can be represented as follows [12], [18]:

Bias:

The expected value of $\hat{\vec{\beta}}$:

$$\begin{aligned}
E(\hat{\vec{\beta}}) &= E((X^{'}X)^{-1}X^{'}\vec{y}) \\
&= E((X^{'}X)^{-1}X^{'}(X\vec{\beta}+\epsilon)) \\
&= E((X^{'}X)^{-1}X^{'}X\vec{\beta}+(X^{'}X)^{-1}X^{'}\epsilon) \\
&= (X^{'}X)^{-1}X^{'}X\vec{\beta}+(X^{'}X)^{-1}X^{'}E(\epsilon) \\
&\text{(Here } (X^{'}X)^{-1}X^{'}X\vec{\beta} \text{ and } (X^{'}X)^{-1}X^{'} \text{ is a matrix of constant.)} \\
&= I\vec{\beta}+0\,(\because (X^{'}X)^{-1}X^{'}X = I, E(\epsilon)=0.) \\
&= \vec{\beta}.
\end{aligned}$$

Thus, the required expected value of $\hat{\vec{\beta}}$ is,

$$E(\hat{\vec{\beta}}) = \vec{\beta}. \tag{2.8}$$

Therefore if the model is accurate, then $\hat{\vec{\beta}}$ is the unbiased estimator of $\vec{\beta}$.

Variance:

$$\begin{aligned}
\mathbf{Var}(\hat{\vec{\beta}}) &= \mathbf{Cov}(\hat{\vec{\beta}},\hat{\vec{\beta}}) \\
&= (X^{'}X)^{-1}X^{'}\mathbf{Cov}(\vec{y},\vec{y})((X^{'}X)^{-1}X^{'})^{'} \\
&= (X^{'}X)^{-1}X^{'}\mathbf{Var}(\vec{y})((X^{'}X)^{-1}X^{'})^{'}.
\end{aligned}$$

$$\begin{aligned}
\mathbf{Var}(\vec{y}) &= E(\vec{y}(\vec{y})^{'}) - E(\vec{y})E(\vec{y})^{'} \\
&= E((X\vec{\beta} + \epsilon)(X\vec{\beta} + \epsilon)^{'}) - x\vec{\beta}(X\vec{\beta})^{'} \\
&= E(X\vec{\beta}(\vec{\beta})^{'}X^{'} + X\vec{\beta}(\vec{\epsilon})^{'} + \vec{\epsilon}(\vec{\beta})^{'}X^{'} + \vec{\epsilon}(\vec{\epsilon})^{'}) - x\vec{\beta}(\vec{\beta})^{'}X^{'} \\
&= X\vec{\beta}(\vec{\beta})^{'}X^{'} + 0 + 0E(\vec{\epsilon}(\vec{\epsilon})^{'})) - X\vec{\beta}(\vec{\beta})^{'}X^{'} \\
&= E((\vec{\beta})^{2}) \\
&= \mathbf{Var}(\vec{\beta}) + E(\vec{\beta})^{2} \\
&= \mathbf{Var}(\vec{\beta}) \\
&= I\sigma^{2}.
\end{aligned}$$

Now,

$$\begin{aligned}
\mathbf{Var}(\hat{\vec{\beta}}) &= (X^{'}X)^{-1}X^{'}I\sigma^{2}((X^{'}X)^{-1}X^{'})^{'} \\
&= \sigma^{2}(X^{'}X)^{-1}(X^{'}X)(X^{'}X)^{-1} \\
&= \sigma^{2}(X^{'}X)^{-1}.
\end{aligned}$$

Therefore,

$$\mathbf{Var}(\hat{\vec{\beta}}) = \sigma^{2}(X^{'}X)^{-1}. \tag{2.9}$$

In equation (2.9) the matrix contains the variances and co-variances of the estimated coefficients where the co-variances are in the off-diagonal elements and the variances of the estimated coefficients are in the diagonal elements [18].

Let $C = (X^{'}X)^{-1}$. Then

$$\begin{aligned}
&\mathbf{Var}(\hat{\vec{\beta}}) = \sigma^{2}c_{ii}, \\
&\mathbf{Cov}(\hat{\vec{\beta}}_{i}, \hat{\vec{\beta}}_{j}) = \sigma^{2}c_{ij}, \\
&\mathbf{Corr}(\hat{\vec{\beta}}_{i}, \hat{\vec{\beta}}_{j}) = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}}.
\end{aligned}$$

2.2.2 THE METHOD OF MAXIMUM LIKELIHOOD ESTIMATION (MLE)

It is considered that, in the regression model (2.5), the errors are independently distributed as well as follow the normal distribution with mean zero and constant variance, σ^2, i.e., $\epsilon \sim N(0, \sigma^2 I)$.

For the errors the normal density function is as follows:

$$f(\epsilon_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}\epsilon_i^2\right), \forall i = 1, 2, \cdots, n \tag{2.10}$$

with the joint density of $\epsilon_1, \epsilon_2, \cdots, \epsilon_n$, the likelihood function can be written as:

$$\begin{aligned} L(\vec{\beta}, \sigma^2) = \prod_{i=1}^{n} f(\epsilon_i) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^2\right) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\epsilon'\epsilon\right) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(\vec{y} - X\vec{\beta})'(\vec{y} - X\vec{\beta})\right). \end{aligned} \tag{2.11}$$

It is known that a log transformation is monotonic, because in original dataset, it preserves the order of the values and it is easy to deal with the log of the likelihood, so $\ln L(\vec{\beta}, \sigma^2)$ is maximized instead of $L(\vec{\beta}, \sigma^2)$. Now taking $\ln$ on both sides of equation (2.11),

$$\begin{aligned} \ln L(\vec{\beta}, \sigma^2) &= -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\vec{y} - X\vec{\beta})'(\vec{y} - X\vec{\beta}) \\ &= -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\vec{y} - X\vec{\beta})^2. \end{aligned} \tag{2.12}$$

Differentiating on both sides of equation (2.12) with respect to $\vec{\beta}$ and σ^2,

$$\begin{aligned} \frac{\partial \ln L(\vec{\beta}, \sigma^2)}{\partial \vec{\beta}} &= \frac{1}{2\sigma^2} 2X'(\vec{y} - X\vec{\beta}) \\ &= \frac{1}{\sigma^2} X'(\vec{y} - X\vec{\beta}). \end{aligned} \tag{2.13}$$

$$\begin{aligned} \frac{\partial \ln L(\vec{\beta}, \sigma^2)}{\partial \vec{\beta}} &= -\frac{n}{2}\frac{1}{2\pi\sigma^2}2\pi - \frac{1}{2}(-1)(\sigma^2)^{-2}(\vec{y} - X\vec{\beta})'(\vec{y} - X\vec{\beta}) \\ &= -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}(\vec{y} - X\vec{\beta})'(\vec{y} - X\vec{\beta}). \end{aligned} \tag{2.14}$$

Now by equating the first order derivative (2.13) and (2.14), the maximum likelihood estimator $\hat{\vec{\beta}}$ and $\hat{\sigma}^2$ given as following:

$$\begin{aligned}
&\left.\frac{\partial \ln L(\vec{\beta},\sigma^2)}{\partial \vec{\beta}}\right|_{\vec{\beta}=\hat{\vec{\beta}}} = 0 \\
\text{or,}\quad &\frac{1}{\sigma^2}X'(\vec{y}-X\vec{\beta}) = 0 \\
\text{or,}\quad &X'\vec{y}-X'X\hat{\vec{\beta}} = 0 \\
\text{or,}\quad &X'X\hat{\vec{\beta}} = X'\vec{y} \\
\text{or,}\quad &(X'X)'X'X\hat{\vec{\beta}} = (X'X)'X'\vec{y} \\
\text{or,}\quad &\hat{\vec{\beta}} = (X'X)'X'\vec{y}. \qquad (2.15)
\end{aligned}$$

$$\begin{aligned}
&\left.\frac{\partial \ln L(\vec{\beta},\sigma^2)}{\partial \sigma^2}\right|_{\sigma^2=\hat{\sigma}^2} = 0 \\
\text{or,}\quad &-\frac{n}{2\hat{\sigma^2}}+\frac{1}{2(\hat{\sigma^2})^2}(\vec{y}-X\vec{\beta})'(\vec{y}-X\vec{\beta}) = 0 \\
\text{or,}\quad &\frac{n}{2\hat{\sigma^2}} = \frac{1}{2(\hat{\sigma^2})^2}(\vec{y}-X\vec{\beta})'(\vec{y}-X\vec{\beta}) \\
\text{or,}\quad &n = \frac{1}{\hat{\sigma}^2}(\vec{y}-X\vec{\beta})'(\vec{y}-X\vec{\beta}) \\
\text{or,}\quad &\hat{\sigma}^2 = \frac{1}{n}(\vec{y}-X\vec{\beta})'(\vec{y}-X\vec{\beta}). \qquad (2.16)
\end{aligned}$$

Since rank$(X) = k$, so $\hat{\vec{\beta}}$ and $\hat{\sigma}^2$ is the required maximum likelihood estimator (m.l.e) of $\vec{\beta}$ and σ^2. The second order partial derivative of $\ln L(\vec{\beta},\sigma^2)$ with respect to $\vec{\beta}$ and σ^2 is as follows:

$$\begin{aligned}
\frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial \vec{\beta}^2} &= -\frac{1}{\sigma^2}X'X. \\
\frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial (\sigma^2)^2} &= \frac{n}{2\sigma^4}-\frac{1}{\sigma^6}(\vec{y}-X\vec{\beta})^2. \\
\frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial \vec{\beta}\partial \sigma^2} &= -\frac{1}{\sigma^4}X'(\vec{y}-X\vec{\beta}). \\
\frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial \sigma^2\partial \vec{\beta}} &= -\frac{n}{2\sigma^2}-\frac{1}{\sigma^4}X(\vec{y}-X\vec{\beta}).
\end{aligned}$$

The Hessian matrix can be written as:

$$H = \begin{bmatrix} \frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial \vec{\beta}^2} & \frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial \vec{\beta} \partial \sigma^2} \\ \frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial (\sigma^2)^2} & \frac{\partial^2 \ln L(\vec{\beta},\sigma^2)}{\partial \sigma^2 \partial \vec{\beta}} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{\sigma^2} X'X & -\frac{1}{\sigma^4} X'(\vec{y} - X\vec{\beta}) \\ \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}(\vec{y} - X\vec{\beta})^2 & -\frac{n}{2\sigma^2} - \frac{1}{\sigma^4} X(\vec{y} - X\vec{\beta}) \end{bmatrix}.$$

The determinant of leading principal minor of order 1 is,

$$D_1(H) = \left| -\tfrac{1}{\sigma^2} X'X \right| < 0.$$

The determinant of leading principal minor of order 2 is,

$$D_2(H) = \begin{vmatrix} -\frac{1}{\sigma^2} X'X & -\frac{1}{\sigma^4} X'(\vec{y} - X\vec{\beta}) \\ \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}(\vec{y} - X\vec{\beta})^2 & -\frac{n}{2\sigma^2} - \frac{1}{\sigma^4} X(\vec{y} - X\vec{\beta}) \end{vmatrix} > 0.$$

Since the leading principal minors are alternative signs, i.e., $D_1(H) < 0$ and $D_2(H) > 0$, so the Hessian H is negative definite at $\vec{\beta} = \hat{\vec{\beta}}$ and $\sigma^2 = \hat{\sigma}^2$. This confirms that the likelihood function is maximized at these values.

After comparing between ordinary least-squared estimator and maximum likelihood estimator, conclude that ordinary least-squared estimator and maximum likelihood estimator are identical and maximum likelihood estimator of $\vec{\beta}$ is also an unbiased estimator of $\vec{\beta}$ [12], [20].

CHAPTER 3

ADVANCED METHODS FOR MODEL INADEQUACIES

3.1 MULTICOLLINEARITY

In multiple regression analysis, the term multicollinearity refers to the presence of a linear relationship among the independent variables. Collinearity indicates two variables that are close perfect linear combinations of one another. Multicollinearity occurs when the regression model includes several variables that are significantly correlated not only with the dependent variable but also to each other.

3.1.1 TYPES OF MULTICOLLINEARITY

Multicollinearity can be divided into two parts. One is **"Data-based multicollinearity"** and another one is **"Structural multicollinearity"**, which are discussed with an example in the following way:

1. **Data-based multicollinearity:** Data-based multicollinearity occurs when there is intercorrelation among the predictor variables in the sample data. This phenomenon is due to the specific dataset and may not accurately represent the characteristics of the overall population. It arises because of a poorly designed experiment by the researchers or because of purely observational data [22].

   Example 3.1.1. *In example 1.2.1 of chapter 1, we have a dataset with information about home sales; in this case, the sale price serves as the dependent variable, while the number of bedrooms and the house's size in square feet serve as the independent variables. Now, assume a dataset where there is a strong correlation between house of the size and the number of the bedroom. Thus, it stands to reason that larger houses also tend to have more bedrooms.*

It is possible to find multicollinearity in a regression model that uses house size and number of bedrooms as independent variables to forecast the selling price of a property. This is due to the fact that the coefficient estimates become unstable when one variable (size, for example) provides information that is repetitive with another variable (bedrooms, for example).

Remark 3.1. *It is possible that this correlation will not hold true after collecting a new sample or additional data.*

2. **Structural multicollinearity:** It occurs when the researcher generates new independent variable from one or more existing variables, for example creating x^3 from x , it is in fact mathematical artifact which leads to multicollinearity [22].

Example 3.1.2. *Consider the following scenario, to fit a polynomial regression model of the following form onto a dataset containing a single independent variable x and a dependent variable y:*

$$y_i = \beta_1 x_i + \beta_2 x_i^3 + \epsilon_i.$$

The independent variables in this model are x and x^3. Nevertheless, multicollinearity problems may arise if the model incorporates both x and x^3; this is due to the strong correlation between the two variables in the majority of datasets.

The design matrix is given below:

$$\begin{bmatrix} 1 & x_1 & x_1^3 \\ 1 & x_1 & x_1^3 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^3 \end{bmatrix}$$

Here, the i^{th} observation is represented by x_i for the independent variable x. There is a structural multicollinearity in the model due to the mathematical relationship between x and x^3.

Because of this, problems like unstable coefficient estimates and inflated standard errors might arise, which inflate the confidence interval of estimated coefficients and make it hard to tell whether the coefficients are significant.

3.1.2 Consequences of Multicollinearity

Multicollinearity is the event of great inter-correlations among the factors in a multiple regression model. Multicollinearity can lead to biased or misleading findings when a researcher tries to determine the best way to use each component in order to predict or understand the response variable in a statistical model. Multicollinearity in regression analysis can have several consequences, including [4], [12]:

1. Multicollinearity inflates the standard errors of the regression coefficients, making them larger than they would be without multicollinearity. As a result, confidence intervals for the coefficients become wider, reducing the precision of the estimates. That is, the findings from a model with multicollinearity may not be trustworthy.

2. Multicollinearity can lead to instability in the estimation of coefficients. Small changes in the data or model specification can result in significant changes in the estimated coefficients, making them difficult to interpret and potentially misleading.

3. In the presence of multicollinearity, the significance tests for individual coefficients may be unreliable. Variables that are actually important predictors of the outcome variable may appear to be statistically insignificant due to multicollinearity. That is, multicollinearity makes some of the significant variables under study to be statistically insignificant.

4. Multicollinearity can lead to misleading interpretations of the relationships between predictor variables and the outcome variable. It becomes challenging to assess the

unique contribution of each predictor variable to the model and obscure the identification of important predictors in the model.

3.1.3 Techniques for Identifying Multicollinearity

Multicollinearity among the variables is examined using different methods. In this study we will discuss the following three methods [4], [12].

1. Pairwise scatterplot
2. Pearson's Correlation Coefficients
3. Variance Inflation Factor

Pairwise scatterplot

A scatterplot is used to observe the relationship between the variables. The scatterplot is a graphical method that signifies the linear relationship between pairs of independent variables. It is important to look for any scatterplots that seem to indicate a linear relationship between pairs of independent variables. It uses dots to represent values for two different variables. The location of each dot on the horizontal and vertical axis denotes values for an individual data point. It is useful to find outliers and observe the patterns between some dimensions.

Pearson's Correlation Coefficients

Pearson's correlation coefficient is a statistical measure of the strength of a linear relationship between paired data. The correlation coefficient is calculated using the formula:

$$r = \frac{n(\sum XY) - (\sum X)(\sum Y)}{\sqrt{(n\sum X^2 - (\sum X)^2)(n\sum Y^2 - (\sum Y)^2)}},$$

where,

r is the correlation coefficient,

n is the number of observations,

X is the first variable in a sample,

Y is the second variable in a sample.

Correlation can take on any value in the range $[-1, 1]$. The sign of the correlation coefficient indicates the direction of the relationship, while the magnitude of the correlation (how close it is to -1 or $+1$) indicates the strength of the relationship.

If the correlation coefficient value is higher with the pairwise variables, it indicates possibility of multicollinearity.

Variance Inflation Factor (VIF)

Variance inflation factor is used to measure how much the variance of the estimated regression coefficient is inflated if the independent variables are correlated. When correlation exists among predictors, the standard error of predictors coefficients will increase and consequently the variance of predictor's coefficients are inflated. That is, in the absence of multicollinearity, the co-variance matrix $\mathbf{Var}(\hat{\vec{\beta}}) = (X^T X)^{-1}\sigma^2$ and the variance of the j^{th} estimator $\hat{\beta}_j, \forall j = 1, 2, \cdots, k$ is written as $\mathbf{Var}(\hat{\vec{\beta}}_j) = [(X^T X)^{-1}]_{jj}\sigma^2)$, where $[\cdots]_{jj}$ indicates the j^{th} diagonal elements of $\mathbf{Var}(\hat{\vec{\beta}})$. Given that the j^{th} predictor X_j is correlated with other predictors $(x_1, x_2, \cdots, x_{j-1}, x_{j+1}, \cdots, x_k)$, then the variance

$$\text{Var}(\hat{\vec{\beta}}_j) = [(X^T X)^{-1}]_{jj}\sigma^2 \times \frac{1}{1 + R_j^2},$$

where

$$VIF_j = \frac{1}{1 + R_j^2}$$

is the variance inflation factor for the variance of $\hat{\vec{\beta}}_{jj}$ and R_j^2 is the coefficient of determination, obtained by regressing x_j against $(x_1, x_2, \cdots, x_{j-1}, x_{j+1}, \cdots, x_k)$.

The value of $VIF_j = 1, \forall j = 1, 2, \cdots, k$ indicates that the independent variables are not correlated to each other. If the value of VIF_j is $1 < VIF_j < 5$, it specifies that the predictor variables are moderately correlated to each other. The challenging value of VIF_j is between 5 to 10 as it specifies the highly correlated variables. If $5 \leq VIF_j \leq 10, \forall j = 1, 2, \cdots, k$, there will be multicollinearity among the predictors in the regression model and $VIF_j > 10$ indicate the regression coefficients are feebly estimated with the presence of multicollinearity [4].

3.1.4 SUGGESTED REMEDY FOR MULTICOLLINEAITY

There are several ways to fix this multicollinearity problem. Some of the techniques are briefly described in the following [14]:

1. **Centering Variables:** Multicollinearity among the first-order, second-order, and higher-order terms for any given predictor variable can be reduced in polynomial regression models by using centered data for the predictor variable's (or variables).

2. **Variable Selection:** Reducing the standard errors of the predicted regression coefficients of the remaining predictor variables and reducing multicollinearity can be achieved by dropping one or more predictor variables from the model. Two significant restrictions exist with this corrective technique.

    - The first issue is that the deleted predictor variables are not directly analyzed.
    - Secondly, the extraneous correlated predictor variables have an effect on the magnitudes of the remaining predictor variables' regression coefficients.

3. **Combine Variables:** It is more efficient to use a single composite variable to reflect the underlying construct than to use numerous associated variables. Use Body Mass Index (BMI) rather than height and weight as an example.

4. **Principal Component Analysis (PCA):** To minimize the amount of variables in a dataset while retaining the maximum amount of variability, one dimensionality reduction technique is Principal Component Analysis (PCA). Principal component analysis does this by converting the input variables into a new set of variables that are linear combinations of the input variables.

   Identifying the directions (or principle components) in which the data differs most is the primary objective of PCA. The data variation that can be explained by the first principal component is the greatest; the variance that can be explained by the second principal component is the second highest; and so on. The relationships between each major component are orthogonal, or uncorrelated.

5. **Two stage least-squares:** In certain economic research, multicollinearity issues can be avoided by estimating the regression coefficients for various predictor variables from various data sets. For instance, in demand study a model is given as

   $$y_i = \beta_0 + \beta_1 x_{i1} + x_{i2} + x_{i3} + \epsilon_i,$$

   where the predictor variables are "income" which is denoted by x_1, "price" which is denoted by x_2, and the response variable is "demand" which is denoted by y_i for i^{th} observations. From cross-section data the coefficient of the predictor variable income x_1 can be estimated and the demand variable can be adjusted as:

   $$y_i^{'} = y_i - \beta_1 x_{i1}.$$

   Then the coefficient of the predictor price β_2, is estimated by regressing the adjusted response variable $y_i^{'}$ on x_2.

6. **Regularization:** Apply regularization methods like Ridge Regression or Lasso Regression to decrease the size and variation of the regression coefficients by adding a penalty term to them.

3.2 THE GENERALIZED LEAST SQUARES ESTIMATION

In this section, the assumptions of the generalized least squares method, its derivation, and generalized least squares estimators for a linear model will be discussed in detail. Consider the following model

$$\begin{aligned} \vec{y} =& X\vec{\beta} + \vec{\epsilon}, \\ E(\epsilon) = 0, & \mathbf{Var}(\epsilon) = \sigma^2 V. \end{aligned} \tag{3.1}$$

In the above model (3.1), $\mathbf{Var}(\epsilon) = \sigma^2 V$, where V is a $n \times n$ non-singular, positive definite and symmetric matrix [12], [21]. It violates the usual assumptions $\mathbf{Var}(\epsilon) = \sigma^2 I$ of the multiple regression model (2.5). In this model the matrix V can be interpreted in the following way [21]:

1. If V is diagonal but unequal variances, then observations $\vec{y}$ are uncorrelated but contain unequal variances.

2. The observations are correlated, when some of the off-diagonal elements of V are nonzero.

In this case the ordinary least-squares estimators $\vec{\beta} = (X'X)^{-1}X'\vec{y}$ is not applicable. Because ordinary least-squares estimates provides unbiased estimates but has more variability, which can be shown as

$$\begin{aligned} E(\vec{\beta}) &= (X'X)^{-1}X'E(\vec{y}) \\ &= (X'X)^{-1}X'X\vec{\beta} \\ &= \vec{\beta}, \end{aligned}$$

$$\begin{aligned} \mathbf{Var}(\vec{\beta}) &= (X'X)^{-1}X'\mathbf{Var}(\vec{y})X(X'X)^{-1} \\ &= \sigma^2(X'X)^{-1}X'VX(X'X)^{-1}. \end{aligned}$$

This situation is defined as "**heteroscedasticity**". So a new approach must be adopted to solve this issue. In order to address this issue, the model can be converted into a new set of observations that agree to the conventional least-squares assumptions. Then following the transformation, the transformed data can be evaluated by ordinary least squares.

3.2.1 Derivation of the Generalized Least squares method for a linear model

Since V is a positive definite, symmetric, so there exists a $n \times n$ non-singular, symmetric matrix K such that [12], [21],

$$K'K = KK = V.$$

So the matrix K can be called the square root of V. Premultiply by K^{-1} on both sides of the model (3.1) yields that,

$$K^{-1}\vec{y} = K^{-1}X\vec{\beta} + K^{-1}\vec{\epsilon}. \tag{3.2}$$

Now define new variables

$$K^{-1}\vec{y} = \vec{z},\ K^{-1}X = B,\ K^{-1}\vec{\epsilon} = \vec{g}.$$

Equation (3.2) can be written as

$$\vec{z} = B\vec{\beta} + \vec{g}, \tag{3.3}$$

which is the required transformed new linear model of the above linear model (3.1).
Now observe that,

$$E(\vec{g}) = E(K^{-1}\epsilon) = K^{-1}E(\vec{\epsilon}) = 0,$$

$$
\begin{aligned}
\mathbf{Var}(\vec{g}) &= [\vec{g} - E(\vec{g})][\vec{g} - E(\vec{g})]^{'} \\
&= E(\vec{g}\vec{g}^{'}) \\
&= E(K^{-1}\vec{\epsilon}(K^{-1}\vec{\epsilon})^{'}) \\
&= K^{-1}E(\vec{\epsilon}(\vec{\epsilon})^{'})(K^{'})^{-1} \\
&= K^{-1}\mathbf{Var}(\vec{\epsilon})K^{-1}(\because K' = K, \mathbf{Var}(\vec{\epsilon}) = E((\vec{\epsilon})^2) - (E(\vec{\epsilon}))^2) \\
&= K^{-1}\sigma^2 V K^{-1} \\
&= \sigma^2 K^{-1} V K^{-1} \\
&= \sigma^2 (K^{-1}K)(KK^{-1}) \\
&= \sigma^2 I.
\end{aligned}
$$

Therefore, the value of mean of the elements of $\vec{g}$ is zero, the elements of $\vec{g}$ has constant variances and are uncorrelated. Since the error $\vec{g}$ of the model (3.3) satisfied the usual assumptions, so ordinary least squares would be applicable. So the least squares function is

$$
\begin{aligned}
S(\vec{\beta}) &= (\vec{g})^{'}\vec{g} \\
&= (K^{-1}\vec{\epsilon})^{'}K^{-1}\vec{\epsilon} \\
&= (\vec{\epsilon})^{'}(K^{-1})^{'}K^{-1}\vec{\epsilon} \\
&= (\vec{\epsilon})^{'}(K^{'})^{-1}K^{-1}\vec{\epsilon} \\
&= (\vec{\epsilon})^{'}(KK)^{-1}\vec{\epsilon}(\because K^{'} = K) \\
&= (\vec{\epsilon})^{'}(V)^{-1}\vec{\epsilon} \\
&= (\vec{y} - X\vec{\beta})^{'}V^{-1}(\vec{y} - X\vec{\beta}). \qquad (3.4)
\end{aligned}
$$

Now differentiate on both sides of equation (3.4) with respect to β yields that,

$$
\begin{aligned}
\frac{\partial S}{\partial \vec{\beta}} &= -X^{'}V^{-1}(\vec{y} - X\vec{\beta}) + ((\vec{y})^{'} - (\vec{\beta})^{'}X^{'})V^{-1}(-X) \\
&= -2X^{'}V^{-1}\vec{y} + 2X^{'}V^{-1}X\vec{\beta},
\end{aligned}
$$

and

$$\frac{\partial^2 S}{\partial(\vec{\beta})^2} = 2X^{'}V^{-1}X,$$

where $\frac{\partial^2 S}{\partial(\vec{\beta})^2}$ is non-negative definite. Then the least squared estimator must satisfy

$$\frac{\partial S}{\partial \vec{\beta}}\Big|_{\hat{\vec{\beta}}} = -2X^{'}V^{-1}\vec{y} + 2X^{'}V^{-1}X\hat{\vec{\beta}} = 0,$$

which simplifies to

$$(X^{'}V^{-1}X)\hat{\vec{\beta}} = X^{'}V^{-1}\vec{y}. \tag{3.5}$$

Equation (3.5) are called the least squares normal equations.

3.2.2 Generalized Least squares estimators for a linear model

Now to find out the least-squares estimator of $\vec{\beta}$, solve the normal equations by multiplying on both sides of equation (3.5) by $(X^{'}V^{-1}X)^{-1}$.

$$\begin{aligned}
&(X^{'}V^{-1}X)^{-1}X^{'}X\hat{\vec{\beta}} = (X^{'}V^{-1}X)^{-1}X^{'}\vec{y} \\
\text{or, } &I\hat{\vec{\beta}} = (X^{'}V^{-1}X)^{-1}X^{'}\vec{y} \\
\text{or, } &\hat{\vec{\beta}} = (X^{'}V^{-1}X)^{-1}X^{'}\vec{y},
\end{aligned}$$

which is the required ordinary least squares estimator (OLSE) of $\vec{\beta}$. Since $\frac{\partial^2 S}{\partial(\vec{\beta})^2}$ is non-negative definite, so $\hat{\vec{\beta}}$ minimize $S(\vec{\beta})$.

Alternatively, after applying OLS to the transformed new linear model (3.3) the

generalized least-squared estimator can be written as

$$\begin{aligned}
\hat{\vec{\beta}} &= (B^{'}B)^{-1}B^{'}\vec{z} \\
&= [(K^{-1}X)^{'}(K^{-1}X)]^{-1}(K^{-1}X)^{'}K^{-1}\vec{y} \\
&= (X^{'}(K^{-1})^{'}K^{-1}X)^{-1}X^{'}(K^{-1})^{'}K^{-1}\vec{y} \\
&= (X^{'}(K^{'})^{-1}K^{-1}X)^{-1}X^{'}(K^{'})^{-1}K^{-1}\vec{y} \\
&= (X^{'}(KK)^{-1}X)^{-1}X^{'}(KK)^{-1}\vec{y}\,(\because K^{'} = K) \\
&= (X^{'}V^{-1}X)^{-1}X^{'}V^{-1}\vec{y}.
\end{aligned}$$

Therefore,

$$\hat{\vec{\beta}} = (X^{'}V^{-1}X)^{-1}X^{'}V^{-1}\vec{y}. \tag{3.6}$$

This equation (3.6) is called the Generalized Least-Squared Estimator (GLSE) of $\vec{\beta}$.

Now it is very easy to prove that $\hat{\vec{\beta}}$ is an unbiased estimator of $\vec{\beta}$. The expected value of GLSE is,

$$\begin{aligned}
E(\hat{\vec{\beta}}) &= E((X^{'}V^{-1}X)^{-1}X^{'}V^{-1}\vec{y}) \\
&= (X^{'}V^{-1}X)^{-1}X^{'}V^{-1}E(\vec{y}) \\
&= (X^{'}V^{-1}X)^{-1}(X^{'}V^{-1}X)\vec{\beta} \\
&= \vec{\beta}.
\end{aligned}$$

Thus this finding demonstrates that GLSE serves as an unbiased estimator of $\vec{\beta}$. The GLSE covariance matrix is provided by

$$\begin{aligned}
\mathbf{Var}(\hat{\vec{\beta}}) &= \mathbf{Var}((X^{'}V^{-1}X)^{-1}X^{'}V^{-1}\vec{y}) \\
&= ((X^{'}V^{-1}X)^{-1}X^{'}V^{-1})\mathbf{Var}(\vec{y})((X^{'}V^{-1}X)^{-1}X^{'}V^{-1})^{'} \\
&= (X^{'}V^{-1}X)^{-1}X^{'}V^{-1}\sigma^2 V(V^{-1}X(X^{'}V^{-1}X)^{-1}) \\
&= \sigma^2(X^{'}V^{-1}X)^{-1}X^{'}V^{-1}V(V^{-1}X(X^{'}V^{-1}X)^{-1}) \\
&= \sigma^2(X^{'}V^{-1}X)^{-1}.
\end{aligned}$$

Therefore, GLSE can be considered the most optimal linear unbiased estimator of $\vec{\beta}$ [12], [21].

3.3 Weighted Least Squares Estimation

When the error ϵ are uncorrelated but have unequal variances, then the covariance matrix of ϵ can be written as:

$$\mathbf{Var}(\vec{\epsilon}) = \sigma^2 V = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_n^2 \end{bmatrix} = \begin{bmatrix} v_{11} & 0 & 0 & \dots & 0 \\ 0 & v_{22} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & v_{nn} \end{bmatrix}.$$

That means,

$$V = (v_{ij})_{n\times n} = \begin{cases} v_{ii} = \sigma_i^2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

where the reciprocal of each variance, σ_i^2, is defined as the weight, which mathematically can be expressed as [2]

$$w_i = \frac{1}{\sigma_i^2}. \tag{3.7}$$

Then consider a diagonal matrix W that contains the these weights in its leading or main diagonal:

$$W = \begin{bmatrix} w_1 & 0 & 0 & \dots & 0 \\ 0 & w_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & w_n \end{bmatrix}.$$

The relation between variance and weight can also be written as:

$$\sigma_i^2 = \frac{1}{w_i}. \tag{3.8}$$

Then V can be expressed by the following way [12]:

$$V = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_n^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{w_1} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{w_2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{w_n} \end{bmatrix} = W^{-1}.$$

Therefore, the relation between variance and weight matrix can be expressed as,

$$V = W^{-1} \text{ or, } W = V^{-1}.$$

Since V is a diagonal matrix, so W is also a diagonal matrix, where diagonal elements of the matrix W are the weights $w_1, w_2, \cdots, w_n$. From equation (3.5), the weighted least-squares normal equations are

$$(X^{'}WX)\hat{\vec{\beta}} = X^{'}W\vec{y}. \tag{3.9}$$

Therefore the weighted least-square estimator is

$$\hat{\vec{\beta}} = (X^{'}WX)^{-1}X^{'}W\vec{y}. \tag{3.10}$$

Alternatively, it is possible to find out the weighted least-squared estimates by transforming the model to a new set of observations. After multiplying each of the observed values for the i^{th} observation (including the 1 for the intercept) by the square root of the

weight for that observation, the transformed set of data [12], [14]:

$$B = \begin{bmatrix} \sqrt{w_1} & X_{11}\sqrt{w_1} & \dots & X_{1k}\sqrt{w_1} \\ \sqrt{w_2} & X_{21}\sqrt{w_2} & \dots & X_{2k}\sqrt{w_2} \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{w_n} & X_{n1}\sqrt{w_n} & \dots & X_{nk}\sqrt{w_n} \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{w_1} & 0 & 0 & \dots & 0 \\ 0 & \sqrt{w_2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sqrt{w_n} \end{bmatrix} \begin{bmatrix} 1 & X_{11} & \dots & X_{1k} \\ 1 & X_{21} & \dots & X_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & \dots & X_{nk} \end{bmatrix} = W^{1/2}X.$$

Thus $B = W^{1/2}X$, or $X = (W^{1/2})^{-1}B$.

$$\vec{z} = \begin{bmatrix} y_1\sqrt{w_1} \\ y_2\sqrt{w_2} \\ \vdots \\ y_n\sqrt{w_n} \end{bmatrix} = \begin{bmatrix} \sqrt{w_1} & 0 & 0 & \dots & 0 \\ 0 & \sqrt{w_2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sqrt{w_n} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = W^{1/2}\vec{y}.$$

Thus $\vec{z} = W^{1/2}\vec{y}$, or $\vec{y} = (W^{1/2})^{-1}\vec{z}$.

$$\vec{g} = \begin{bmatrix} \epsilon_1\sqrt{w_1} \\ \epsilon_2\sqrt{w_2} \\ \vdots \\ \epsilon_n\sqrt{w_n} \end{bmatrix} = \begin{bmatrix} \sqrt{w_1} & 0 & 0 & \dots & 0 \\ 0 & \sqrt{w_2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sqrt{w_n} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} = W^{1/2}\vec{\epsilon}.$$

Thus $\vec{g} = W^{1/2}\vec{\epsilon}$, or $\vec{\epsilon} = (W^{1/2})^{-1}\vec{g}$.

Now after applying this transformation in equation (3.1), a new transformed linear model will be found which is given as below:

$$(W^{1/2})^{-1}\vec{z} = (W^{1/2})^{-1}B\vec{\beta} + (W^{1/2})^{-1}\vec{g}$$

$$\text{or, } W^{1/2}(W^{1/2})^{-1}\vec{z} = W^{1/2}(W^{1/2})^{-1}B\vec{\beta} + W^{1/2}(W^{1/2})^{-1}\vec{g}$$

$$\text{or, } \vec{z} = B\vec{\beta} + \vec{g}.$$

Now observed that, the errors have zero expectation in the above new transformed model, i.e.,

$$\begin{aligned} E(\vec{g}) &= E(W^{1/2}\vec{\epsilon}) \\ &= W^{1/2}E(\vec{\epsilon}) \\ &= 0\,(\because E(\vec{\epsilon}) = 0), \end{aligned}$$

and the covariance matrix of errors of the new transformed model is

$$\begin{aligned} \mathbf{Var}(\vec{g}) &= E[\vec{g} - E(\vec{g})][\vec{g} - E(\vec{g})]' \\ &= E(gg') \\ &= E(W^{1/2}\vec{\epsilon}(W^{1/2}\vec{\epsilon})') \\ &= E(W^{1/2}\vec{\epsilon}(\vec{\epsilon})'(W^{1/2})') \\ &= W^{1/2}E(\vec{\epsilon}(\vec{\epsilon})')W^{1/2} \\ &= E(W^{1/2}\mathbf{Var}(\vec{\epsilon})W^{1/2}) \\ &= E(W^{1/2}\sigma^2 V W^{1/2}) \\ &= E(W^{1/2}\sigma^2 W^{-1}W^{1/2}) \\ &= \sigma^2 W^{1/2}(W^{1/2}W^{1/2})^{-1}W^{1/2} \\ &= \sigma^2 W^{1/2}(W^{1/2})^{-1}(W^{1/2})^{-1}W^{1/2} \\ &= \sigma^2 I. \end{aligned}$$

The weighted least square estimate of $\vec{\beta}$,

$$\begin{aligned} \hat{\vec{\beta}} &= (B'B)^{-1}B'\vec{z} \\ &= ((W^{1/2}X)'(W^{1/2}X))^{-1}(W^{1/2}X)'(W^{1/2}\vec{y}) \\ &= (X'(W^{1/2})'(W^{1/2}X))^{-1}(X'(W^{1/2})')(W^{1/2}\vec{y}) \\ &= (X'W^{1/2}W^{1/2}X)^{-1}(X'W^{1/2})(W^{1/2}\vec{y}) \\ &= (X'WX)^{-1}(X'W)\vec{y}. \end{aligned}$$

Thus, $\hat{\vec{\beta}} = (X'WX)^{-1}(X'W)\vec{y}$ is the required weighted least-squared estimates of $\vec{\beta}$.

3.3.1 SELECTING THE WEIGHT FOR WLS REGRESSION

OLS does not discriminate between the quality of the observations, giving equal weight to each, irrespective of whether they are good or poor guides to the location of the line. Thus, it may be concluded that if we can find a way of assigning more weight to high-quality observations and less to the unreliable ones, we are likely to obtain a better fit. In other words, our estimators for coefficients will be more efficient. WOLS works by incorporating extra non-negative constants (weights) associated with each data point into the fitting criterion.

Suppose the true relationship is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \forall i = 1, 2, ..., n,$$

where $E[\epsilon_i] = 0$ and $\mathbf{Var}(\epsilon_i) = \sigma_i^2$, which is a heteroscedastic model.

In ordinary least squares, the estimated coefficients provide the regression equation that minimizes $SSE = \sum e_i^2$. In weighted least squares (WLS), the estimated equation minimizes $\sum w_i e_i^2$ where w_i is a weight given to the i^{th} observation. The object is to minimize the sum of the squares of the random factors of the estimated residuals. If the weights are all the same constant, then we have ordinary least squares (OLS) regression. However, if the structure of the data suggests unequal weights are appropriate, then it would be inappropriate to ignore the regression weights.

Error Variance Unknown

Utilizing weighted least squares with weights w_i is a very simple process if the variances σ_i^2 are either known or can be determined by a proportional constant. However, in reality, these variances, σ_i^2, are rarely known, therefore requiring the use of estimated

variances. The use of some possible variance and standard deviation functions are as follows [14]:

1. If a residual plot against a predictor exhibits a megaphone shape, then regress the absolute values of the residuals against that predictor. The resulting fitted values of this regression are estimates of σ_i.

2. If a residual plot against the fitted values exhibits a megaphone shape, then regress the absolute values of the residuals against the fitted values. The resulting fitted values of this regression are estimates of σ_i.

3. If a residual plot of the squared residuals against a predictor exhibits an upward trend, then regress the squared residuals against that predictor. The resulting fitted values of this regression are estimates of σ_i^2.

4. If a residual plot of the squared residuals against the fitted values exhibits an upward trend, then regress the squared residuals against the fitted values. The resulting fitted values of this regression are estimates of σ_i^2.

5. If the predictors are discrete or continuous with many replications for each x_i value, then arrange the dataset in descending to ascending order and cluster the datasets with replications for each x_i values. Find out the mean values of the predictors and sample variance of the response variables for each cluster. Then regress the sample variances, S_y^2, against the average values, $\bar{x}$, i.e.,

$$S_y^2 \sim \gamma_0 + \gamma_1 \bar{x},$$

where γ_0 and γ_1 are the intercept and slope of this regression model respectively. After that substituting each x_i value into the above equation will give the estimate of the variance $\hat{\sigma}_i^2$ (which is the fitted value of the above variance function) of the

corresponding observation y_i and the required weights will be the reciprocal of this $\hat{\sigma}_i^2$, which mathematically can be written as,

$$w_i = \frac{1}{\hat{\sigma}_i^2},$$

or, if $\hat{v}_i$ is fitted value from standard deviation function, then the above equation can be written as

$$w_i = \frac{1}{\hat{v}_i}.$$

After using one of these methods to estimate the weights, w_i, these weights can be used in weighted least squares regression model.

CHAPTER 4

GAUSS NEWTON ITERATIVE METHOD (GNIM) FOR NONLINEAR LEAST SQUARES ESTIMATION

4.1 THE NONLINEAR REGRESSION MODEL

Regression analysis in which the relationship between the independent and dependent variables is not linear, that means, it does not follow a linear relationship with the unknown parameters is considered a nonlinear regression model. For instance, the model

$$y = \theta_1 e^{-\theta_2 x} + \epsilon \tag{4.1}$$

is not linear with respect to unknown parameters θ_1 and θ_2. In general, the nonlinear regression model can be written as,

$$\vec{y} = f(x, \vec{\theta}) + \epsilon, \tag{4.2}$$

where $\vec{\theta}$ is a $p \times 1$ vector of parameters and for any nonlinear regression model, $f(x, \vec{\theta})$ is the expectation function, which is a nonlinear function of the parameters. Consider the above nonlinear model (4.1), the expectation function is $f(x, \vec{\theta}) = \theta_1 e^{-\theta_2 x}$. Then the derivatives of this expectation function with respect to parameters θ_1 and θ_2 are

$$\frac{\partial f(x, \vec{\theta})}{\partial \theta_1} = e^{-\theta_2 x},$$
$$\frac{\partial f(x, \vec{\theta})}{\partial \theta_2} = -\theta_1 x e^{-\theta_2 x}.$$

Since the derivatives are functions of the unknown parameters θ_1 and θ_2, the above stated model (4.1) is a nonlinear [12].

4.1.1 DIFFERENCE BETWEEN LINEAR AND NONLINEAR REGRESSION MODEL

The key differences between the linear and nonlinear regression model are given below:

- In linear regression model, the expectation function is a linear function of the parameters, while in nonlinear regression model this expectation function is nonlinear function with respect to parameters.

- In linear regression model, the derivatives of expectation function with respect to the parameters are not functions of the unknown parameters.

  For instance, consider the regression model (2.3), where the expectation function is as follows:

$$\begin{aligned} f(x, \vec{\beta}) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \\ &= \beta_0 + \sum_{j=1}^{k} \beta_j x_j. \end{aligned}$$

  Now the derivatives with respect to the parameters $\beta_1, \beta_2, \cdots, \beta_k$ are

$$\frac{\partial f(x, \vec{\beta})}{\partial \beta_0} = 1, \ \frac{\partial f(x, \vec{\beta})}{\partial \beta_1} = x_1, \ \frac{\partial f(x, \vec{\beta})}{\partial \beta_2} = x_2, \cdots, \frac{\partial f(x, \vec{\beta})}{\partial \beta_k} = x_k.$$

  In general,

$$\frac{\partial f(x, \vec{\beta})}{\partial \beta_j} = x_j, \forall j = 1, 2, \cdots k.$$

  It is clear that the derivatives are not functions of the parameters β. So the regression model (2.3) is a linear regression model.

  On the other hand, at least one of the expectation function's derivatives with respect to the parameters in a nonlinear regression model is dependent on at least one of the parameters, which has already been discussed in section 4.1.

4.1.2 Assumptions

In the following, the assumptions of the nonlinear model will be briefly discussed. Similar to the linear model, the nonlinear model assumes [14] that

1. The expected value of error terms, $E(\epsilon_i) = 0, \forall i$.

2. The variances of error terms are homogeneous, $\mathbf{Var}(\epsilon_i) = \sigma^2, \forall i$.

3. The random error terms are uncorrelated, that means, the error terms are independent, $\mathbf{Cov}(\epsilon_i, \epsilon_j) = 0, \forall i, j$.

4. The error terms are normally distributed.

4.1.3 TYPES OF NONLINEAR REGRESSION MODEL

The classification of the nonlinear regression model, which will be briefly discussed, is given below. There are two types of nonlinear regression models. They are

1. Parametric nonlinear regression model

2. Non-parametric nonlinear regression model

Parametric Nonlinear Regression:

If the dependent and independent variables can be related by a particular non-linear mathematical function with unknown constants, then the regression model can be called parametric non-linear. An exponential function, for instance, can be used to model the relationship between a country's population and time. The polynomial regression, logistic regression, exponential regression, power regression and so on are the common examples of parametric nonlinear regression model.

Non-Parametric Nonlinear Regression:

Unlike parametric non-linear regression, non-parametric non-linear regression does not presume that a particular mathematical function can express the relationship between the dependent and independent variables. In non-parametric linear regression machine

learning algorithms are used to learn the association between the dependent and independent variables. Kernel smoothing, local polynomial regression, nearest neighbor regression and so on are the common examples of non-parametric regression model.

4.1.4 THE NONLINEAR LEAST SQUARES METHOD FOR PARAMETER ESTIMATION

Consider a sample of n observations, where the regressors are $x_{i1}, x_{i2}, \cdots, x_{ip}$, for $i = 1, 2, \cdots, n$ and the response are y_i, for $i = 1, 2, \cdots, n$.

Now consider the nonlinear regression model (4.2), where $x_i^{'} = (x_{i1}, x_{i2}. \cdots, x_{ip})$, for $i = 1, 2, \cdots, n$. The sum of squares error function is

$$S(\vec{\theta}) = \sum_{i=1}^{n} [y_i - f(x_i, \vec{\theta})]^2. \tag{4.3}$$

After differentiating on both sides of the equation (4.3) with respect to each element of $\vec{\theta}$, i.e., $\theta_1, \theta_2, \cdots, \theta_p$ and equating the resulting equations to zero yields that,

$$\frac{\partial S(\vec{\theta})}{\partial \vec{\theta}} = 2 \sum_{i=1}^{n} [y_i - f(x_i, \vec{\theta})] \left[\frac{\partial f(x_i, \vec{\theta})}{\partial \theta_j} \right]_{\vec{\theta}=\hat{\vec{\theta}}} = 0, \forall j = 1, 2, \cdots, p. \tag{4.4}$$

After simplifying equation (4.4), the normal equations are

$$\sum_{i=1}^{n} [y_i - f(x_i, \vec{\theta})] \left[\frac{\partial f(x_i, \vec{\theta})}{\partial \theta_j} \right]_{\vec{\theta}=\hat{\vec{\theta}}} = 0, \forall j = 1, 2, \cdots, p. \tag{4.5}$$

In the normal equations (4.5) of the nonlinear regression model (4.1), the expectation function is a nonlinear function and the derivatives would be the functions of unknown parameters. Thus the nonlinear equations (4.5) are not in a closed-form to be solved as was the case for linear regression. In this case it is very difficult to solve the normal equations [12]. Iterative methods (such as Newton method, Gauss-Newton iteration method, method of steepest descent, Marquardt's method, direct search, etc) must be applied to find values of the parameters of $\theta_1, \theta_2, \cdots, \theta_n$.

4.2 GAUSS NEWTON ITERATIVE METHOD FOR NONLINEAR REGRESSION

A method widely used for nonlinear regression is linearization of the nonlinear function followed by the Gauss-Newton iterative method of parameter estimation. The Gauss-Newton method, so named in honor of mathematicians Carl Friedrich Gauss and Isaac Newton, is an iterative optimization strategy used to reduce the residuals in a nonlinear least squares problem. When an initial estimate of the parameters is available, it works especially well [24]. By iteratively improving the parameter estimations, the Gauss-Newton approach converges to values that minimize the sum of squared residuals. The method is computationally efficient since it uses a linear approximation of the nonlinear model at each iteration. Consider the following model

$$y_i = f(\vec{x}_i, \vec{\theta}) + \vec{\epsilon}. \tag{4.6}$$

Linearization is accomplished by a Taylor series expansion of $f(\vec{x}_i, \vec{\theta})$ about the point $\vec{\theta}^{(0)} = [\theta_1^{(0)}, \theta_2^{(0)}, ..., \theta_p^{(0)}]$, which is called the starting values of the parameters, with only the linear terms retained [12].

Thus, this produces

$$f(\vec{x}_i, \vec{\theta}) = f(\vec{x}_i, \vec{\theta}^{(0)}) + \sum_{j=1}^{p} \left[\frac{\partial f(\vec{x}_i, \vec{\theta})}{\partial \theta_j} \right]_{\vec{\theta}=\vec{\theta}^{(0)}} (\theta_j - \theta_j^{(0)}). \tag{4.7}$$

Now, consider

$$\begin{aligned} f_i^{(0)} &= f(\vec{x}_i, \vec{\theta}^{(0)}), \\ z_{ij}^{(0)} &= \left[\frac{\partial f(\vec{x}_i, \vec{\theta}^{(0)})}{\partial \theta_j} \right]_{\vec{\theta}=\vec{\theta}^{(0)}}, \\ \beta_j^{(0)} &= \theta_j - \theta_j^{(0)}. \end{aligned}$$

Now using equation (4.7) and the above new defining variable in nonlinear model (4.6), the

nonlinear regression model (4.6) can be written as

$$y_i = f_i^{(0)} + \sum_{j=1}^{p} \beta_j^{(0)} z_{ij}^{(0)} + \epsilon_i$$

$$\text{or, } y_i - f_i^{(0)} = \sum_{j=1}^{p} \beta_j^{(0)} z_{ij}^{(0)} + \epsilon_i. \tag{4.8}$$

That is, a linear regression model has been developed. This developed linear regression model (4.8) can be written in the following way:

$$\begin{bmatrix} y_1 - f_1^{(0)} \\ y_2 - f_2^{(0)} \\ \vdots \\ y_n - f_n^{(0)} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{p} \beta_j^{(0)} z_{1j}^{(0)} \\ \sum_{j=1}^{p} \beta_j^{(0)} z_{2j}^{(0)} \\ \vdots \\ \sum_{j=1}^{p} \beta_j^{(0)} z_{nj}^{(0)} \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\text{or, } \begin{bmatrix} y_1 - f_1^{(0)} \\ y_2 - f_2^{(0)} \\ \vdots \\ y_n - f_n^{(0)} \end{bmatrix} = \begin{bmatrix} \beta_1^{(0)} z_{11}^{(0)} + \beta_2^{(0)} z_{12}^{(0)} + ... + \beta_p^{(0)} z_{1p}^{(0)} \\ \beta_1^{(0)} z_{21}^{(0)} + \beta_2^{(0)} z_{22}^{(0)} + ... + \beta_p^{(0)} z_{2p}^{(0)} \\ \vdots \\ \beta_1^{(0)} z_{n1}^{(0)} + \beta_2^{(0)} z_{n2}^{(0)} + ... + \beta_p^{(0)} z_{np}^{(0)} \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\text{or, } \begin{bmatrix} y_1 - f_1^{(0)} \\ y_2 - f_2^{(0)} \\ \vdots \\ y_n - f_n^{(0)} \end{bmatrix} = \begin{bmatrix} z_{11}^{(0)} & z_{12}^{(0)} & \dots & z_{1p}^{(0)} \\ z_{21}^{(0)} & z_{22}^{(0)} & \dots & z_{2p}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1}^{(0)} & z_{n2}^{(0)} & \dots & z_{np}^{(0)} \end{bmatrix} \begin{bmatrix} \beta_1^{(0)} \\ \beta_2^{(0)} \\ \vdots \\ \beta_p^{(0)} \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}, \tag{4.9}$$

where

$$\vec{y}^{(0)} = \begin{bmatrix} y_1 - f_1^{(0)} \\ y_2 - f_2^{(0)} \\ \vdots \\ y_n - f_n^{(0)} \end{bmatrix}, \; Z^{(0)} = \begin{bmatrix} z_{11}^{(0)} & z_{12}^{(0)} & \dots & z_{1p}^{(0)} \\ z_{21}^{(0)} & z_{22}^{(0)} & \dots & z_{2p}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1}^{(0)} & z_{n2}^{(0)} & \dots & z_{np}^{(0)} \end{bmatrix}, \; \vec{\beta^{(0)}} = \begin{bmatrix} \beta_1^{(0)} \\ \beta_2^{(0)} \\ \vdots \\ \beta_p^{(0)} \end{bmatrix}, \; \vec{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

Equation (4.9) can be written as

$$\vec{y}^{(0)} = Z^{(0)}\vec{\beta}^{(0)} + \vec{\epsilon}. \tag{4.10}$$

Hence, the estimated value of $\beta^{(0)}$ is

$$\hat{\vec{\beta}}^{(0)} = ((Z^{(0)})'Z^{(0)})^{-1}(Z^{(0)})'\vec{y}^{(0)}. \tag{4.11}$$

Since $\vec{\beta}^{(0)} = \vec{\theta} - \vec{\theta}^{(0)}$, the revised estimates of $\vec{\theta}$ can be defined as:

$$\vec{\theta} = \hat{\vec{\beta}}^{(0)} + \vec{\theta}^{(0)} = \hat{\vec{\theta}}^{(1)},$$

where sometimes $\hat{\vec{\beta}}^{(0)}$ is called the **vector of increments**. Similarly, $\hat{\vec{\theta}}^{(1)}$ will play the same role like initial estimates $\vec{\theta}^{(0)}$ and after plugging this in equation (4.7), another set of revised estimates would be found, say $\hat{\vec{\theta}}^{(2)}$, and so forth. i.e.,

$$\begin{aligned}
\hat{\vec{\beta}}^{(1)} + \hat{\vec{\theta}}^{(1)} &= \hat{\vec{\theta}}^{(2)},\\
\hat{\vec{\beta}}^{(2)} + \hat{\vec{\theta}}^{(2)} &= \hat{\vec{\theta}}^{(3)},\\
\hat{\vec{\beta}}^{(3)} + \hat{\vec{\theta}}^{(3)} &= \hat{\vec{\theta}}^{(4)},\\
&\vdots\\
\hat{\vec{\beta}}^{(k)} + \hat{\vec{\theta}}^{(k)} &= \hat{\vec{\theta}}^{(k+1)}.
\end{aligned}$$

Therefore, in general at the k^{th} iteration it can be written as

$$\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + ((Z^{(k)})'Z^{(k)})^{-1}(Z^{(k)})'(\vec{y} - \vec{f}^{(k)}), \tag{4.12}$$

where

$$\begin{aligned}
\vec{Z}^{(k)} &= \left[z_{ij}^{(k)}\right],\\
\vec{f}^{(k)} &= \begin{bmatrix} f_1^{(k)} & f_2^{(k)} & \dots & f_n^{(k)} \end{bmatrix}',\\
\hat{\vec{\theta}}^{(k)} &= \begin{bmatrix} \theta_1^{(k)} & \theta_2^{(k)} & \dots & \theta_p^{(k)} \end{bmatrix}'.
\end{aligned}$$

This iterative process continues until there is no noticeable difference between two consecutive estimated coefficients and this convergence can be measured by the following convergence metric [12], i.e.,

$$\frac{\hat{\theta}_j^{(k+1)} - \hat{\theta}_j^{(k)}}{\hat{\theta}_j^{(k)}} < \delta, \forall j = 1, 2, ..., p, \tag{4.13}$$

where δ is a tiny number, say 0.000001. The residual sum of squares, $S(\hat{\vec{\theta}}^{(k)})$, should be determined after every iteration to make sure that the reduction in its value has been achieved.

4.2.1 Application of the Gauss-Newton to a Logistic growth model

In this section, the "Logistic growth model" with three parameters $\vec{\theta} = (\theta_1, \theta_2, \theta_3)$, and one regressor variable x is considered. An attempt to fit the nonlinear model to data is made by applying the *Gauss-Newton iterative Method (GNIM)* derived in Section 4.2. The nonlinear model is given in following Example 4.2.1.

Example 4.2.1. *The following nonlinear function is a* ***Logistic Growth Model*** *[12]. The model is*

$$y = \frac{\theta_1}{1 + \theta_2 e^{-\theta_3 x}} + \epsilon, \tag{4.14}$$

where $\epsilon \sim N(E(\epsilon) = 0, \mathbf{Var}(\epsilon) = \sigma^2$*, consider here* σ^2 *is a constant), will be fitted to the paired data [16] by applying the* Gauss-Newton Iterative Method (GNIM).

Table 4.1: Nonlinear least square dataset, where x represents "predictor variable" and y represents "response variable ".

Obs. No.	Predictor, x	Response, y
1	1	5.308
2	2	7.24
3	3	9.638
4	4	12.866
5	5	17.069
6	6	23.192
7	7	31.443
8	8	38.558
9	9	50.156
10	10	62.948
11	11	75.995
12	12	91.972

Solution 4.2.1. *The model is*

$$y = \frac{\theta_1}{1 + \theta_2 e^{-\theta_3 x}} + \epsilon = f(x, \vec{\theta}) + \epsilon, \tag{4.15}$$

where $f(x, \vec{\theta})$ *is the* ***expectation function*** *for the above nonlinear regression model. In general form model* (4.15) *can be expressed as,*

$$y_i = f(x_i, \vec{\theta}) + \epsilon_i, \forall i = 1, 2, ..., n, \tag{4.16}$$

where

$$\vec{\theta} = (\theta_1 \quad \theta_2 \quad \theta_3)',$$

$$\textit{and} \quad f(x_i, \vec{\theta}) = \frac{\theta_1}{1 + \theta_2 e^{\theta_3 x_i}}, \forall i = 1, 2, ..., n. \tag{4.17}$$

Equation (4.16) *can be written as,*

$$\epsilon_i = y_i - f(x_i, \vec{\theta}).$$

Then the sum of squares of the residuals is,

$$S(\vec{\theta}) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} [y_i - f(x_i, \vec{\theta})]^2.$$

Equation (4.10) *can be obtained after converting the given nonlinear regression model to linear regression model using linearization method which has already been discussed in the above section 4.2.*

In equation (4.10) *for this problem, for* $n = 12$*,* $p = 3$ *(p is the number of parameters), the Jacobian matrix consists of the following components.*

$$\begin{aligned} z_{i1} &= \frac{\partial f(x_i, \vec{\theta})}{\partial \theta_1} = \frac{1}{1 + \theta_2 e^{-\theta_3 x_i}}, \\ z_{i2} &= \frac{\partial f(x_i, \vec{\theta})}{\partial \theta_2} = \frac{-\theta_1 e^{-\theta_3 x_i}}{1 + \theta_2 e^{-\theta_3 x_i^2}}, \\ z_{i3} &= \frac{\partial f(x_i, \vec{\theta})}{\partial \theta_3} = \frac{\theta_1 \theta_2 x_i}{e^{\theta_3 x_i} \left(1 + \theta_2 e^{\theta_3 x_i}\right)^2}, \end{aligned} \tag{4.18}$$

where $\forall i = 1, 2, \cdots, n$*. In general the Jacobian matrix can be expressed by*

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ \vdots & \vdots & \vdots \\ z_{n1} & z_{n2} & z_{n3} \end{bmatrix} = (z_{ij}), \forall i = 1, 2, \ldots n, \forall j = 1, 2, \ldots, p, \tag{4.19}$$

where "i" represents "observations" and "j" represents "parameters". For this particular "Logistic growth" model (4.16) *the Jacobian matrix is given below:*

$$Z = \begin{bmatrix} \frac{\partial f(x_1, \vec{\theta})}{\partial \theta_1} & \frac{\partial f(x_1, \vec{\theta})}{\partial \theta_2} & \frac{\partial f(x_1, \vec{\theta})}{\partial \theta_3} \\ \frac{\partial f(x_2, \vec{\theta})}{\partial \theta_1} & \frac{\partial f(x_2, \vec{\theta})}{\partial \theta_2} & \frac{\partial f(x_2, \vec{\theta})}{\partial \theta_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial f(x_n, \vec{\theta})}{\partial \theta_1} & \frac{\partial f(x_n, \vec{\theta})}{\partial \theta_2} & \frac{\partial f(x_n, \vec{\theta})}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{1+\theta_2 e^{-\theta_3 x_1}} & \frac{-\theta_1 e^{-\theta_3 x_1}}{1+\theta_2 e^{-\theta_3 x_1^2}} & \frac{\theta_1 \theta_2 x_1}{e^{\theta_3 x_1}\left(1+\theta_2 e^{\theta_3 x_1}\right)^2} \\ \frac{1}{1+\theta_2 e^{-\theta_3 x_2}} & \frac{-\theta_1 e^{-\theta_3 x_2}}{1+\theta_2 e^{-\theta_3 x_2^2}} & \frac{\theta_1 \theta_2 x_2}{e^{\theta_3 x_2}\left(1+\theta_2 e^{\theta_3 x_2}\right)^2} \\ \vdots & \vdots & \vdots \\ \frac{1}{1+\theta_2 e^{-\theta_3 x_n}} & \frac{-\theta_1 e^{-\theta_3 x_n}}{1+\theta_2 e^{-\theta_3 x_n^2}} & \frac{\theta_1 \theta_2 x_n}{e^{\theta_3 x_n}\left(1+\theta_2 e^{\theta_3 x_n}\right)^2} \end{bmatrix}.$$

The residual vector is given by

$$\vec{y}-\vec{f}=\begin{bmatrix} y_1-f_1 \\ y_2-f_2 \\ \vdots \\ y_n-f_n \end{bmatrix}=\begin{bmatrix} y_1-f(x_1,\vec{\theta}) \\ y_2-f(x_2,\vec{\theta}) \\ \vdots \\ y_n-f(x_n,\vec{\theta}) \end{bmatrix}=\begin{bmatrix} y_1-\frac{\theta_1}{1+\theta_2 e^{-\theta_3 x_1}} \\ y_2-\frac{\theta_1}{1+\theta_2 e^{-\theta_3 x_2}} \\ \vdots \\ y_n-\frac{\theta_1}{1+\theta_2 e^{-\theta_3 x_n}} \end{bmatrix}. \tag{4.20}$$

and the Gauss-Newton recursion equation is given as follows:

$$\hat{\vec{\theta}}^{(k+1)}=\hat{\vec{\theta}}^{(k)}+((Z^{(k)})'Z^{(k)})^{-1}(Z^{(k)})'(\vec{y}-\vec{f}^{(k)}),\forall k=1,2,...,$$

where "k" represents the number of iterations.

The computation process of GNIM algorithm is summarized step by step in the following:

Step 1: *Select an initial approximation for* $\vec{\theta}^{(0)}=(\theta_1^{(0)}=200,\theta_2^{(0)}=50.50,\theta_3^{(0)}=0.3035)$ *in equation* (4.17)*. Set the maximum number of iteration* $nsim=5$*.*

Step 2: *Simultaneously compute* z_{i1}*,* z_{i2} *and* z_{i3}*,* $\forall i=1,2,\ldots,n$ *using equation* (4.18)*.*

Step 3: *Estimate the residual* $(\vec{y}-\vec{f}^{(k)})$ *using* (4.20) *at each iteration* $k=1,2,\ldots,nsim$*.*

Step 4: *Estimate the parameter* $\vec{\theta}^{(k+1)}$ *using Gauss-Newton recursion formula* (4.12) *at each iteration* $k=1,2,\ldots,nsim$*.*

Computation-Output: 4.2.1.

1. The Jacobian matrix at each iteration $k = 1, 2, \ldots, 5$ is given below.

$$Z^{(1)} = \begin{pmatrix} z_{i1}^{(1)} & z_{i2}^{(1)} & z_{i3}^{(1)} \\ 0.02739643 & -0.1064444 & 5.173672 \\ 0.03713290 & -0.1428295 & 13.884311 \\ 0.05015122 & -0.1902956 & 27.747654 \\ 0.06741403 & -0.2511493 & 48.827924 \\ 0.09005560 & -0.3273545 & 79.554485 \\ 0.11932851 & -0.4198082 & 122.427365 \\ 0.15648039 & -0.5272880 & 179.399912 \\ 0.20253898 & -0.6452245 & 250.886424 \\ 0.25800785 & -0.7647597 & 334.536740 \\ 0.32252396 & -0.8728683 & 424.253081 \\ 0.39459329 & -0.9543105 & 510.221431 \\ 0.47155749 & -0.9954631 & 580.607611 \end{pmatrix}, Z^{(2)} = \begin{pmatrix} z_{i1}^{(2)} & z_{i2}^{(2)} & z_{i3}^{(2)} \\ 0.02712164 & -0.1054408 & 5.175333 \\ 0.03674397 & -0.1414367 & 13.884221 \\ 0.04960607 & -0.1883965 & 27.741087 \\ 0.06665893 & -0.2486183 & 48.811538 \\ 0.08902487 & -0.3240800 & 79.533784 \\ 0.11794687 & -0.4157340 & 122.432351 \\ 0.15466962 & -0.5224754 & 179.511922 \\ 0.20023049 & -0.6399255 & 251.274730 \\ 0.25516115 & -0.7594711 & 335.492756 \\ 0.31914783 & -0.8683186 & 426.195139 \\ 0.39076232 & -0.9513357 & 513.636512 \\ 0.46741446 & -0.9947773 & 585.917580 \end{pmatrix},$$

$$Z^{(3)} = \begin{pmatrix} z_{i1}^{(3)} & z_{i2}^{(3)} & z_{i3}^{(3)} \\ 0.02711662 & -0.1054282 & 5.175637 \\ 0.03673681 & -0.1414187 & 13.884936 \\ 0.04959597 & -0.1883715 & 27.742364 \\ 0.06664488 & -0.2485846 & 48.813641 \\ 0.08900559 & -0.3240362 & 79.537241 \\ 0.11792091 & -0.4156798 & 122.438251 \\ 0.15463544 & -0.5224126 & 179.522396 \\ 0.20018674 & -0.6398592 & 251.293595 \\ 0.25510698 & -0.7594108 & 335.526071 \\ 0.31908333 & -0.8682776 & 426.251184 \\ 0.39068883 & -0.9513289 & 513.724571 \\ 0.46733467 & -0.9948165 & 586.045310 \end{pmatrix}, Z^{(4)} = \begin{pmatrix} z_{i1}^{(4)} & z_{i2}^{(4)} & z_{i3}^{(4)} \\ 0.02711658 & -0.1054282 & 5.175642 \\ 0.03673674 & -0.1414187 & 13.884948 \\ 0.04959588 & -0.1883714 & 27.742382 \\ 0.06664474 & -0.2485844 & 48.813666 \\ 0.08900539 & -0.3240359 & 79.537273 \\ 0.11792062 & -0.4156794 & 122.438293 \\ 0.15463505 & -0.5224121 & 179.522463 \\ 0.20018622 & -0.6398588 & 251.293720 \\ 0.25510631 & -0.7594104 & 335.526317 \\ 0.31908251 & -0.8682775 & 426.251650 \\ 0.39068788 & -0.9513292 & 513.725381 \\ 0.46733361 & -0.9948174 & 586.046586 \end{pmatrix},$$

$$Z^{(5)} = \begin{pmatrix} z_{i1}^{(5)} & z_{i2}^{(5)} & z_{i3}^{(5)} \\ 0.02711658 & -0.1054282 & 5.175642 \\ 0.03673674 & -0.1414187 & 13.884948 \\ 0.04959588 & -0.1883714 & 27.742382 \\ 0.06664474 & -0.2485844 & 48.813666 \\ 0.08900539 & -0.3240359 & 79.537273 \\ 0.11792062 & -0.4156794 & 122.438293 \\ 0.15463505 & -0.5224121 & 179.522463 \\ 0.20018622 & -0.6398588 & 251.293721 \\ 0.25510631 & -0.7594104 & 335.526319 \\ 0.31908250 & -0.8682775 & 426.251654 \\ 0.39068787 & -0.9513292 & 513.725387 \\ 0.46733360 & -0.9948174 & 586.046596 \end{pmatrix}.$$

2. *The parameter estimates at each iteration* $k = 1, 2, \ldots, 5$ *is given below.*

$$\hat{\vec{\theta}} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix} = \begin{pmatrix} k=1 & k=2 & k=3 & k=4 & k=5 \\ 194.164167 & 196.138915 & 196.18578 & 196.18626 & 196.18626 \\ 48.604481 & 49.082809 & 49.09158 & 49.09164 & 49.09164 \\ 0.314152 & 0.313582 & 0.31357 & 0.31357 & 0.31357 \end{pmatrix}.$$

3 *The error margin (EM) over consecutive iterations is given by* $EM^{(k+1)} = |\hat{\vec{\theta}}^{(k+1)} - \hat{\vec{\theta}}^{(k)}|$ *for convergence of the algorithm is given below.*

$$EM^{(k+1)} = \left| \begin{bmatrix} \hat{\theta}_1^{(k+1)} \\ \hat{\theta}_2^{(k+1)} \\ \hat{\theta}_3^{(k+1)} \end{bmatrix} - \begin{bmatrix} \hat{\theta}_1^{(k)} \\ \hat{\theta}_2^{(k)} \\ \hat{\theta}_3^{(k)} \end{bmatrix} \right| = \begin{pmatrix} k=1 & k=2 & k=3 & k=4 & k=5 \\ 5.835833 & 1.974748 & 0.046865 & 0.00048 & 0 \\ 1.895519 & 0.478328 & 0.008771 & 0.00006 & 0 \\ 0.010652 & 0.00057 & 0.000012 & 0 & 0 \end{pmatrix}.$$

4. *It is easy to form the error margin matrix that the algorithm converges after 5 iterations. Hence, the* ***Maximum Likelihood Estimate*** *for the parameter vector* $\vec{\theta}$ *is given by*

$$\hat{\vec{\theta}} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix} = \begin{bmatrix} 196.18626 \\ 49.09164 \\ 0.31357 \end{bmatrix}.$$

Remark 4.1. *The final iterative result derived from the GNIM in nonlinear regression can be considered as the maximum likelihood estimates (MLE's) under specific conditions.*

The iterative optimization process known as the GNIM is applied to resolve nonlinear least squares problems. The GNIM attempts to reduce the sum of squared residuals between the predicted and observed values.

If the errors in the model are considered to be independent and identically distributed (i.i.d.) as well as normally distributed with a mean of zero and constant variance in the context of nonlinear regression, minimizing the sum of squared residuals is equivalent to maximizing the likelihood function. The GNIM's estimates can be considered as MLE's under these assumptions [23].

Testing for Multicollinearity

The variance inflation factors among the columns of jacobian matrix z_{i1}, z_{i2}, and z_{i3} (which are the partial derivatives of the model function with respect to the parameters) at each iteration $k = 1, 2, \ldots, 5$ is given below:

$$\begin{bmatrix} VIF_{z_{i1}} \\ VIF_{z_{i2}} \\ VIF_{z_{i3}} \end{bmatrix} = \begin{pmatrix} \text{k=1} & \text{k=2} & \text{k=3} & \text{k=4} & \text{k=5} \\ 360.6293 & 372.24835 & 372.47772 & 372.48094 & 372.48097 \\ 43.6020 & 43.23491 & 43.22759 & 43.22747 & 43.22747 \\ 536.2783 & 545.20741 & 545.38129 & 545.38371 & 545.38373 \end{pmatrix}.$$

In the above table, the VIF (Variance Inflation Factor) values for each iteration indicate the presence of multicollinearity in the given data. According to the discussion in subsection 3.1.3, VIF measures how much the variance of an estimated regression coefficient is increased due to multicollinearity in the model.

In the final iteration, VIF for z_{i1} is 372.48097, indicating a high degree of multicollinearity between z_{i1} and the other independent variables in the model. VIF for z_{i2} is 43.22747, which is lower than z_{i1} but still suggests some multicollinearity. VIF for z_{i3} is

545.38373, indicating a high degree of multicollinearity involving z_{i3} and the other independent variables.

Generally, VIF values above 10 are considered indicative of multicollinearity. In this case, both z_{i1}, z_{i2}, and z_{i3} have VIF values well above these thresholds, suggesting that multicollinearity is likely influencing the regression results.

CHAPTER 5

WEIGHTED GAUSS-NEWTON ITERATIVE METHOD (WGNIM)

To get the most accurate parameter estimations, it's important to use weighted least squares regression to give each data point the weight it deserves. When estimating the parameters, the weighted least squares fitting criterion is used to minimize

$$S(\vec{\beta}) = \sum_{i=1}^{n} w_i[y_i - f(\vec{x}_i, \hat{\vec{\beta}})]. \tag{5.1}$$

In order to achieve optimal outcomes that reduce uncertainty in the parameter estimators, it is necessary to assign weights, denoted as w_i, to the unknown parameters. These weights should be inversely proportional to the variances associated with each combination of predictor variable values, i.e.,

$$w_i \propto \frac{1}{\sigma_i^2}. \tag{5.2}$$

5.1 Derivation of Weighted Gauss-Newton Iterative Method (WGNIM)

The issue is that the actual variances of the data points, which determine these optimal weights, are rarely known. It is necessary to substitute the estimated weights. The optimality properties linked to known weights are no longer strictly applicable when estimated weights are utilized. Weights, if they can be computed with enough precision, can greatly improve the value of estimated coefficients as compared to the results that would be obtained if all data points were weighted equally.

In this section, the method of weighted least squares, which was discussed in section 3.3 will be applied to extend the *Gauss-Newton iterative method*, which was already discussed in section 4.2.

Consider the model (4.6) of section 4.2. Recall the Gauss-Newton iterative method, where, after applying the linearization method to the function $f(\vec{x}_i, \vec{\theta})$, the linear regression model (4.10) has already been found in section 4.2.

Now to apply weighted least square method define the following new variables:

$$\begin{aligned} B = W^{1/2}Z^{(0)} \quad &\Rightarrow \quad Z^{(0)} = (W^{1/2})^{-1}B, \\ \vec{z} = W^{1/2}\vec{y}^{(0)} \quad &\Rightarrow \quad \vec{y}^{(0)} = (W^{1/2})^{-1}\vec{z}, \\ \vec{g} = W^{1/2}\vec{\epsilon} \quad &\Rightarrow \quad \vec{\epsilon} = (W^{1/2})^{-1}\vec{g}. \end{aligned}$$

After applying the above transformation in equation (4.10) yields that,

$$\begin{aligned} (W^{1/2})^{-1}\vec{z} &= (W^{1/2})^{-1}B\vec{\beta} + (W^{1/2})^{-1}\vec{g} \\ \text{or, } W^{1/2}(W^{1/2})^{-1}\vec{z} &= W^{1/2}(W^{1/2})^{-1}B\vec{\beta} + W^{1/2}(W^{1/2})^{-1}\vec{g} \\ \text{or, } \vec{z} &= B\vec{\beta} + \vec{g}, \end{aligned} \tag{5.3}$$

which is the required new linear regression model. Then the weighted least square estimate of $\vec{\beta}$ is,

$$\begin{aligned} \hat{\vec{\beta}} &= (B'B)^{-1}B'\vec{z} \\ &= ((W^{1/2}Z^{(0)})'(W^{1/2}Z^{(0)}))^{-1}(W^{1/2}Z^{(0)})'(W^{1/2}\vec{y}^{(0)}) \\ &= ((Z^{(0)})'(W^{1/2})'W^{1/2}Z^{(0)})^{-1}(Z^{(0)})'(W^{1/2})'W^{1/2}\vec{y}^{(0)} \\ &= ((Z^{(0)})'WZ^{(0)})^{-1}(Z^{(0)})'W\vec{y}^{(0)}. \end{aligned}$$

Therefore, the required estimate of $\vec{\beta}$ using GNIM with weights is

$$\hat{\vec{\beta}} = ((Z^{(0)})'WZ^{(0)})^{-1}(Z^{(0)})'W\vec{y}^{(0)}. \tag{5.4}$$

Since $\vec{\beta}^{(0)} = \vec{\theta} - \vec{\theta}^{(0)}$, the revised estimates of θ can be defined as:

$$\vec{\theta} = \hat{\vec{\beta}}^{(0)} + \vec{\theta}^{(0)} = \hat{\vec{\theta}}^{(1)},$$

where sometimes $\hat{\vec{\beta}}^{(0)}$ is called the **vector of increments**. Similarly, $\hat{\vec{\theta}}^{(1)}$ will play the same role like initial estimates $\vec{\theta}^{(0)}$ and after plugging this in equation (4.7), another set of revised

estimates will be obtained, say $\hat{\vec{\theta}}^{(2)}$, and so forth. i.e.,

$$\begin{aligned}
\hat{\vec{\beta}}^{(1)} + \hat{\vec{\theta}}^{(1)} &= \hat{\vec{\theta}}^{(2)}, \\
\hat{\vec{\beta}}^{(2)} + \hat{\vec{\theta}}^{(2)} &= \hat{\vec{\theta}}^{(3)}, \\
\hat{\vec{\beta}}^{(3)} + \hat{\vec{\theta}}^{(3)} &= \hat{\vec{\theta}}^{(4)}, \\
&\vdots \\
\hat{\vec{\beta}}^{(k)} + \hat{\vec{\theta}}^{(k)} &= \hat{\vec{\theta}}^{(k+1)}.
\end{aligned}$$

Therefore, in general at the k^{th} iteration the Weighted Gauss-Newton recursion formula can be written as

$$\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + ((Z^{(k)})' W Z^{(k)})^{-1} (Z^{(k)})' W (\vec{y} - \vec{f}^{(k)}), \tag{5.5}$$

where

$$\begin{aligned}
\vec{Z}^{(k)} &= \left[z_{ij}^{(k)} \right], \\
\vec{f}^{(k)} &= \begin{bmatrix} f_1^{(k)} & f_2^{(k)} & \dots & f_n^{(k)} \end{bmatrix}', \\
\hat{\vec{\theta}}^{(k)} &= \begin{bmatrix} \theta_1^{(k)} & \theta_2^{(k)} & \cdots & \theta_p^{(k)} \end{bmatrix}',
\end{aligned}$$

and the diagonal weight matrix is given by

$$W = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}.$$

This iterative process continues until there is no noticeable difference between two consecutive estimated coefficients, and this convergence can be measured by the following convergence metric,

$$\frac{\hat{\theta}_j^{(k+1)} - \hat{\theta}_j^{(k)}}{\hat{\theta}_j^{(k)}} < \delta, \forall j = 1, 2, ..., p, \tag{5.6}$$

where δ is a tiny number, say 0.000001. The residual sum of squares, $S(\hat{\vec{\theta}}^k)$, should be determined after every iteration to make sure that the reduction in its value has been achieved.

Remark 5.1. *Equation* (5.5) *is the required recursion formula for the* ***"Weighted Gauss-Newton Iterative Method (WGNIM)"****. In this process, throughout the iterations, the weights remain constant. The weighted sum of squared residuals is minimized by employing these initial weights. More precisely, WGNIM takes into account heteroscedasticity using a predetermined weighting scheme and uses fixed weights that are determined before iterations start.*

In the rare circumstance that the estimated coefficients are significantly different from the estimated regression coefficients obtained by ordinary least squares, it is generally recommended to iterate the weighted least squares process. This involves reestimating the variance or standard deviation function using the residuals from the weighted least squares fit and then obtaining revised weights. Most often, the estimated regression coefficients can be stabilized with just one or two iterations. This iterative technique is often referred to as iteratively reweighted least squares [14].

At iteration k, the computed weighted residuals can be defined by

$$r_i^{(k)} = y_i - f(x_i, \vec{\theta}^{(k)}). \tag{5.7}$$

Recalculate the weights by using these residuals (5.7) after each iteration and the inverse of the squared residuals is a commonly used method, which is defined as follows:

$$w_i^{(k+1)} = \frac{1}{(r_i^{(k)})^2}.$$

In subsequent iterations, the influence of observations with large residuals is diminished as a result of these weights. Then generate the latest weight matrix $W^{(k+1)}$ using the updated weights and formulate the weighted Gauss-Newton recursion formula for the next iteration,

i.e., the extended form of (5.4) is

$$\begin{aligned}\hat{\vec{\beta}} &= ((Z^{(0)})'W^{(0)}Z^{(0)})^{-1}(Z^{(0)})'W^{(0)}\vec{y}^{(0)} \\ &= ((Z^{(0)})'W^{(0)}Z^{(0)})^{-1}(Z^{(0)})'W^{(0)}(\vec{y} - \vec{f}^{(0)}),\end{aligned} \tag{5.8}$$

and the extended form of Weighted Gauss-Newton recursion formula (5.5) is

$$\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + ((Z^{(k)})'W^{(k)}Z^{(k)})^{-1}(Z^{(k)})'W^{(k)}(\vec{y} - \vec{f}^{(k)}), \tag{5.9}$$

which is called **"Reweighted Gauss-Newton Iterative Method (RGNIM)"**.

Remark 5.2. *In **"Reweighted Gauss-Newton Iterative Method (RGNIM)"** at each iteration, this algorithm recalculates the weights. The residuals from the prior iteration are used to modify the weights. This is the key difference between the **"Weighted Gauss-Newton Iterative Method (WGNIM)"** and **"Reweighted Gauss-Newton Iterative Method (RGNIM)"**.*

5.2 Application of the Reweighted Gauss-Newton Iterative Method (RGNIM) to a Logistic growth model

In this section, the "Logistic growth model" with three parameters, $\vec{\theta} = (\theta_1, \theta_2, \theta_3)$, and one regressor variable x is considered, which have already discussed in example 4.2.1. In this section, the *Reweighted Gauss-Newton Iterative Method (RGNIM)* is applied to attempt to fit the nonlinear model to the data, which has already been derived in section 5.1.

Example 5.2.1. *Recall the nonlinear model (4.14) of example 4.2.1 is*

$$y = \frac{\theta_1}{1 + \theta_2 e^{-\theta_3 x}} + \epsilon,$$

where $\epsilon \sim N(E(\epsilon) = 0, \mathbf{Var}(\epsilon) = \sigma^2)$, σ is a constant, will be fitted to the paired data given in Table 4.1 by applying the Reweighted Gauss-Newton Iterative Method (RGNIM).

Solution 5.2.1. *The model is*

$$y = \frac{\theta_1}{1 + \theta_2 e^{-\theta_3 x}} + \vec{\epsilon} = f(x, \vec{\theta}) + \vec{\epsilon}, \tag{5.10}$$

where $f(x, \vec{\theta})$ *is the* ***expectation function*** *for the nonlinear regression model. In general form*

$$y_i = f(x_i, \vec{\theta}) + \vec{\epsilon_i}, \forall i = 1, 2, ..., n, \tag{5.11}$$

where "i*" represents the number of observation. In equation* (5.11)*,*

$$f(x_i, \vec{\theta}) = \frac{\theta_1}{1 + \theta_2 e^{-\theta_3 x_i}}, \forall i = 1, 2, ..., n, \tag{5.12}$$

$$\vec{\theta} = (\theta_1 \quad \theta_2 \quad \theta_3)',$$

$$\textit{and } \vec{\epsilon} = (\epsilon_1 \ \ \epsilon_2 \ \ \cdots \ \ \epsilon_n)'.$$

Equation (5.11) *can be written as*

$$\epsilon_i = y_i - f(x_i, \vec{\theta}).$$

Then sum of squares of the residuals,

$$S(\vec{\theta}) = \sum_{i=1}^{n} w_i \epsilon_i^2 = \sum_{i=1}^{n} w_i [y_i - f(x_i, \vec{\theta})]^2.$$

Now equations (4.18) *and* (4.20) *which have already described in example 4.2.1 of chapter 4 will be used for the jacobian matrix and residual vector respectively, and for this problem Reweighted Gauss-Newton recursion formula 5.9 will be applied.*

In example 4.2.1 of chapter 4, high level of multicollinearity were observed between the columns (z_{i1}*,* z_{i2}*, and* z_{i3}*) of the jacobian matrix using the GNIM. Here an attempt will be made to apply Reweighted Gauss-Newton Iterative method in a different new way to alleviate this multicollinearity. According to the transformed linear model* (4.10)*, regress (*$y_i - f_i$*) against the columns of the jacobian matrix* z_{i1}*,* z_{i2}*, and* z_{i3}*. This regression can be denoted by* m_1*, i.e.,*

$$m_1 = lm((y_i - f_i^{(k)}) \sim z_{i1}^{(k)} + z_{i2}^{(k)} + z_{i3}^{(k)}).$$

The residual of this m_1 regression is to be regressed against the predicted value of m_1 regression which is denoted by m_2, i.e.,

$$m_2 = lm(\textit{residual}(m_1) \sim \textit{fitted value}(m_1)).$$

The inverse of the square of the residual extracted from this m_2 regression will be desired weight, i.e.,

$$w_i^{(k)} = \frac{1}{(\textit{residual}(m_2))^2}.$$

For this problem like GNIM consider the following same starting values:

$$\theta_1^{(0)} = 200,\ \theta_2^{(0)} = 50.50,\ \theta_3^{(0)} = 0.3035.$$

The computation process of RGNIM are described in the following steps:

Step 1: *Select an initial approximate value for $\vec{\theta}^{(k)} = \vec{\theta}^{(0)} = (\theta_1^{(0)} = 200, \theta_2^{(0)} = 50.50,$ $\theta_3^{(0)} = 0.3035)$, set the maximum number of simulation $nsim = 5$, and $k = 0$.*

Step 2: *Estimate the residual vector $\vec{r}^{(k)} = (y_i - f_i^{(k)})$ using equation* (4.20).

Step 3: *Simultaneously compute z_{i1}, z_{i2} and z_{i3}, $\forall i = 1, 2, \ldots, n$ using equation* (4.18).

Step 4: *Regress $(y_i - f_i)$ against z_{i1}, z_{i2}, and z_{i3}, i.e.,*

$$m_1 = lm((y_i - f_i^{(k)}) \sim z_{i1}^{(k)} + z_{i2}^{(k)} + z_{i3}^{(k)}).$$

Step 5: *Find the residuals and fitted values from model m_1, then regress residual of m_1 against fitted values of m_1, i.e.,*

$$m_2 = lm(\textit{residual}(m_1) \sim \textit{fitted value}(m_1)).$$

Step 6: *Extract the residuals of model m_2 and take the inverse of the squared of these residuals of m_2, i.e.,*

$$w_i^{(k)} = \frac{1}{(\textit{residual}(m_2))^2}.$$

Step 7: *Make the matrix of weights,* $W^{(k)} = diag(w_1^{(k)}, w_2^{(k)}, \cdots, w_n^{(k)})$.

Step 8: *Calculate* $\hat{\vec{\beta}}^{(k)}$ *using equation* (5.8).

Step 9: *Update* $\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + \hat{\vec{\beta}}^{(k)}$ *using Reweighted Gauss-Newton recursion formula 5.9.*

Step 10: *If* $\frac{(\hat{\theta}_j^{(k+1)} - \hat{\theta}_j^{(k)})}{\hat{\theta}_j^{(k)}} < 0.000001$, *then stop.*

Step 11: *Otherwise, set* $k = k + 1$.

Then, go to the ***step 2***.

Repeat steps 2 to 11 until convergence.

The R code of this algorithm is provided in Appendix A.5.

Computation-Output: 5.2.1.

1. *The Jacobian at each iteration* $k = 1, 2, \ldots, 5$ *is given below.*

$$Z^{(1)} = \begin{pmatrix} z_{i1}^{(1)} & z_{i2}^{(1)} & z_{i3}^{(1)} \\ 0.02783648 & -0.1069375 & 5.126645 \\ 0.03781900 & -0.1437948 & 13.787211 \\ 0.05119287 & -0.1919392 & 27.605035 \\ 0.06895717 & -0.2537029 & 48.650663 \\ 0.09228625 & -0.3310261 & 79.347926 \\ 0.12246966 & -0.4246850 & 122.157903 \\ 0.16077642 & -0.5331830 & 178.927761 \\ 0.20822191 & -0.6514877 & 249.861625 \\ 0.26524347 & -0.7701307 & 332.284685 \\ 0.33134562 & -0.8755062 & 419.722824 \\ 0.40484452 & -0.9521273 & 502.100954 \\ 0.48287323 & -0.9867484 & 567.663612 \end{pmatrix}, Z^{(2)} = \begin{pmatrix} z_{i1}^{(2)} & z_{i2}^{(2)} & z_{i3}^{(2)} \\ 0.02783316 & -0.1069259 & 5.126688 \\ 0.03781432 & -0.1437788 & 13.787285 \\ 0.05118634 & -0.1919175 & 27.605129 \\ 0.06894816 & -0.2536741 & 48.650809 \\ 0.09227398 & -0.3309891 & 79.348296 \\ 0.12245328 & -0.4246394 & 122.159000 \\ 0.16075504 & -0.5331298 & 178.930730 \\ 0.20819481 & -0.6514301 & 249.868654 \\ 0.26521026 & -0.7700750 & 332.299329 \\ 0.33130654 & -0.8754607 & 419.749929 \\ 0.40480058 & -0.9521014 & 502.145825 \\ 0.48282620 & -0.9867495 & 567.730382 \end{pmatrix},$$

$$Z^{(3)} = \begin{pmatrix} z_{i1}^{(3)} & z_{i2}^{(3)} & z_{i3}^{(3)} \\ 0.02783310 & -0.1069257 & 5.126689 \\ 0.03781423 & -0.1437785 & 13.787286 \\ 0.05118621 & -0.1919171 & 27.605131 \\ 0.06894799 & -0.2536736 & 48.650812 \\ 0.09227375 & -0.3309884 & 79.348304 \\ 0.12245297 & -0.4246385 & 122.159022 \\ 0.16075464 & -0.5331288 & 178.930787 \\ 0.20819430 & -0.6514291 & 249.868787 \\ 0.26520964 & -0.7700739 & 332.299605 \\ 0.33130581 & -0.8754599 & 419.750439 \\ 0.40479975 & -0.9521010 & 502.146668 \\ 0.48282532 & -0.9867496 & 567.731637 \end{pmatrix}, Z^{(4)} = \begin{pmatrix} z_{i1}^{(4)} & z_{i2}^{(4)} & z_{i3}^{(4)} \\ 0.02783310 & -0.1069257 & 5.126689 \\ 0.03781423 & -0.1437785 & 13.787286 \\ 0.05118621 & -0.1919171 & 27.605131 \\ 0.06894799 & -0.2536736 & 48.650813 \\ 0.09227375 & -0.3309884 & 79.348304 \\ 0.12245297 & -0.4246385 & 122.159022 \\ 0.16075464 & -0.5331288 & 178.930788 \\ 0.20819430 & -0.6514290 & 249.868789 \\ 0.26520963 & -0.7700739 & 332.299608 \\ 0.33130580 & -0.8754599 & 419.750444 \\ 0.40479974 & -0.9521010 & 502.146677 \\ 0.48282531 & -0.9867496 & 567.731650 \end{pmatrix},$$

$$Z^{(5)} = \begin{pmatrix} z_{i1}^{(5)} & z_{i2}^{(5)} & z_{i3}^{(5)} \\ 0.02783310 & -0.1069257 & 5.126689 \\ 0.03781423 & -0.1437785 & 13.787286 \\ 0.05118621 & -0.1919171 & 27.605131 \\ 0.06894799 & -0.2536736 & 48.650813 \\ 0.09227375 & -0.3309884 & 79.348304 \\ 0.12245297 & -0.4246385 & 122.159022 \\ 0.16075464 & -0.5331288 & 178.930788 \\ 0.20819430 & -0.6514290 & 249.868789 \\ 0.26520963 & -0.7700739 & 332.299608 \\ 0.33130580 & -0.8754599 & 419.750444 \\ 0.40479974 & -0.9521010 & 502.146677 \\ 0.48282531 & -0.9867496 & 567.731651 \end{pmatrix}.$$

2. *The weighted matrix, W, for each iteration $k = 1, 2, \ldots 5$ is as follows (The R code of Appendix A.5 implements the iterative nature of these weights):*

$$W^{(1)} = \begin{pmatrix}
47.13146 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 57.91882 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 360.5215 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 19.9913 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 4.952285 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 2459.5 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.9334985 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 1.779182 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 58.92851 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 5.687531 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 2.874516 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.0000 & 0.000000 & 0.0 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 19.97923
\end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix}
46.73522 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 57.7657 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 357.7139 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 19.90868 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 4.940467 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 2366.467 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.933605 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 1.780787 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 58.40217 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 5.674605 & 0.000000 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 2.874444 & 0.00000 \\
0.00000 & 0.0000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 20.07336
\end{pmatrix}$$

$$W^{(3)} = \begin{pmatrix}
46.72802 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 57.763 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 357.6605 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 19.90714 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 4.940249 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 2364.824 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.9336066 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 1.780818 & 0.00000 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 58.39232 & 0.000000 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 5.674369 & 0.00000 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 2.87444 & 0.0000 \\
0.00000 & 0.000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.00000 & 20.0751
\end{pmatrix}$$

$$W^{(4)} = \begin{pmatrix}
46.72796 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 57.76299 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 357.6598 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 19.90713 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 4.940247 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 2364.812 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.9336065 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 1.780818 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 58.39221 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 5.674367 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 2.874439 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 20.07512
\end{pmatrix}$$

$$W^{(5)} = \begin{pmatrix}
46.72796 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 57.76298 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 357.6598 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 19.90713 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 4.940247 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 2364.812 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.9336065 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 1.780818 & 0.00000 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 58.39221 & 0.000000 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 5.674367 & 0.000000 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 2.874439 & 0.00000 \\
0.00000 & 0.00000 & 0.0000 & 0.00000 & 0.000000 & 0.000 & 0.0000000 & 0.000000 & 0.00000 & 0.000000 & 0.000000 & 20.07512
\end{pmatrix}$$

3. The parameter estimates at each iteration $k = 1, 2, \ldots, 5$ *is given below.*

$$\hat{\vec{\theta}} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix} = \begin{pmatrix}
k\text{=}1 & k\text{=}2 & k\text{=}3 & k\text{=}4 & k\text{=}5 \\
189.4434788 & 189.4669859 & 189.4674275 & 189.4674324 & 189.4674325 \\
47.9405883 & 47.9461738 & 47.9462774 & 47.9462784 & 47.9462784 \\
0.3167857 & 0.3167797 & 0.3167796 & 0.3167796 & 0.3167796
\end{pmatrix}$$

4 The error margin (EM) over consecutive iterations is given by $EM^{(k+1)} = |\hat{\vec{\theta}}^{(k+1)} - \hat{\vec{\theta}}^{(k)}|$ *for convergence of the algorithm is given below.*

$$EM^{(k+1)} = \left| \begin{bmatrix} \hat{\theta}_1^{(k+1)} \\ \hat{\theta}_2^{(k+1)} \\ \hat{\theta}_3^{(k+1)} \end{bmatrix} - \begin{bmatrix} \hat{\theta}_1^{(k)} \\ \hat{\theta}_2^{(k)} \\ \hat{\theta}_3^{(k)} \end{bmatrix} \right| = \begin{pmatrix}
k\text{=}1 & k\text{=}2 & k\text{=}3 & k\text{=}4 & k\text{=}5 \\
10.5565212 & 0.0235071 & 0.0004416 & 0.0000049 & 0.0000001 \\
2.5594117 & 0.0055855 & 0.0001036 & 0.000001 & 0.000000 \\
0.0132857 & 0.000006 & 0.0000001 & 0.000000 & 0.000000
\end{pmatrix}$$

5. *It is easy to from the error margin matrix that the algorithm converges after 5 iterations. Hence, the* ***Maximum Likelihood Estimate*** *for the parameter vector* $\vec{\theta}$ *is given by*

$$\hat{\vec{\theta}} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix} = \begin{bmatrix} 189.4674325 \\ 47.9462784 \\ 0.3167796 \end{bmatrix}$$

Comparison of three iterative methods for the stated particular data and the corresponding **"Logistic Growth Model"** is given below:

Table 5.1: Comparison of three iterative methods.

Methods	β_1	β_2	β_3	$\mathbf{MS_{res}}$
GNIM	196.18626	49.09164	0.31357	0.2874747
WGNIM	189.46743	47.94628	0.31678	0.3060210
Levenberg-Marquardt (using "nls" function)	196.18626	49.09164	0.31357	0.2874747

Table 5.1 compares three iterative strategies for the aforementioned particular dataset and the corresponding "Logistic Growth Model." It takes a closer look at the estimated parameters β_1, β_2, and β_3, and how each method yields the mean squared residuals (MS_{res}). This comparison shows how the three methods are comparable and how they differ in terms of the mean squared residuals and parameter estimates. While the RGNIM produces somewhat different parameter estimates and a greater mean squared residual than the other two approaches, the GNIM and the nls function (where the Levenberg-Marquardt method was applied) produce similar results in the end. For this particular problem, GNIM is better than RGNIM, and the "nls" (using the Levenberg-Marquardt method) function which is flexible for any initial approximate value, whereas GNIM and RGNIM produce singular jacobian matrices for any starting approximate value.

From the above discussion it is clear that, multicollinearity still exists after applying the RGNIM to this problem. More detailed work will be done in the future to address this issue.

CHAPTER 6

APPLICATION OF THE ITERATIVE NONLINEAR REGRESSION METHODS

Using data for ultrasonic calibration, this application shows how to build a nonlinear regression model. As a case study, it shows how to fit a nonlinear model and utilize weighted fits to address the issue of non-constant standard deviations for the errors, also referred to as heterogeneous variances for the errors.

6.1 DESCRIPTION OF DATASETS

There is a predictor variable and a response variable in the ultrasonic reference block data. The ultrasonic calibration block data is typically composed of measurements that are obtained during the calibration of ultrasonic equipment that is utilized for the evaluation of materials and non-destructive testing (NDT). The process of calibration includes confirming that the ultrasonic equipment accurately measures distances and flaws in the material. The calibration block is often made from a standardized material, such as steel or aluminum.

The term **"metal distance"** is the measurement of the distance between the surface of the metal (or any other substance being examined) and a particular target point inside of it. This point could serve as a recognizable reference marker, a flaw, or another notable internal characteristic.

The signal that the ultrasonic transducer receives back after emitting ultrasonic waves into a material is referred to as the **"ultrasonic response"**. This reaction is essentially the reflected sound of the ultrasonic waves that have already penetrated the material and reflected off its many internal characteristics, including imperfections, interfaces, and barriers [25].

The **"metal distance"** serves as the **"predictor variable"**, while the **"ultrasonic response"** serves as the **"response variable"**. Dan Chwirut, a scientist at NIST, furnished

the stated data [15]. The datasets are listed in Table 6.1.

Table 6.1: Ultrasonic reference block datasets, where the predictor, x represents "Metal Distance", and the response, y represents "Ultrasonic response".

No.	Predictor (x)	Response (y)	No.	Predictor (x)	Response (y)	No.	Predictor (x)	Response (y)
1	0.5	92.9	19	3	13.12	37	5.75	3.75
2	1	57.1	20	0.75	59.9	38	3	11.81
3	1.75	31.05	21	3	14.62	39	0.75	54.7
4	3.75	11.5875	22	1.5	32.9	40	2.5	23.7
5	5.75	8.025	23	6	5.44	41	4	11.55
6	0.875	63.6	24	3	12.56	42	0.75	61.3
7	2.25	21.4	25	6	5.44	43	2.5	17.7
8	3.25	14.25	26	1.5	32	44	4	8.74
9	5.25	8.475	27	3	13.95	45	0.75	59.2
10	0.75	63.8	28	0.5	75.8	46	2.5	16.3
11	1.75	26.8	29	2	20	47	4	8.62
12	2.75	16.4625	30	4	10.42	48	0.5	81
13	4.75	7.125	31	0.75	59.5	49	6	4.87
14	0.625	67.3	32	2	21.67	50	3	14.62
15	1.25	41	33	5	8.55	51	0.5	81.7
16	2.25	21.15	34	0.75	62	52	2.75	17.17
17	4.25	8.175	35	2.25	20.2	53	0.5	81.3
18	0.5	81.5	36	3.75	7.76	54	1.75	28.9

6.2 Data Analysis of Results

To identify the pattern of the data, generate a scatter plot of the given ultrasonic reference block dataset.

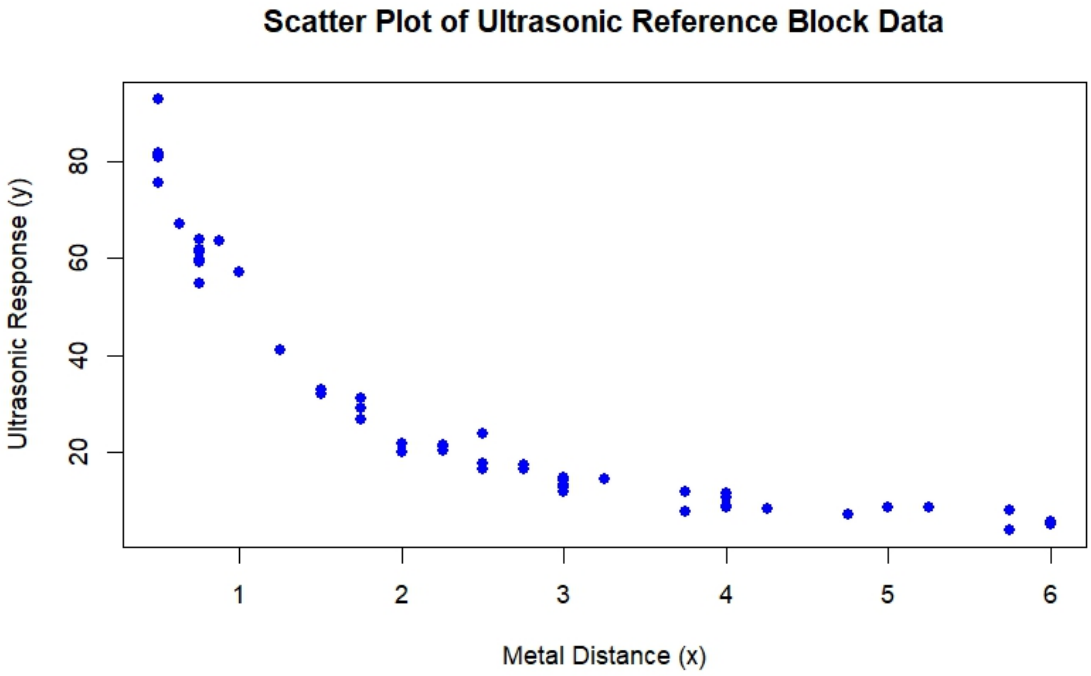


Figure 6.1: Scatter plot of ultrasonic calibration dataset

The above graphic demonstrates a pattern of exponential decay, which implies that an exponential function could potentially serve as a suitable model for the given dataset. It is indispensable to choose perfect initial values for the parameters to find out the estimated coefficients of a model.

The following theoretical model for the response variable (y) and predictor variable (x) will try to fit the particular dataset described in Table 6.1.

Remark 6.1. *The objective of this investigation is to assess the GNIM and the WGNIM for ultrasonic calibration data. The primary objective of this investigation is to evaluate the efficacy of these iterative methods for managing heteroscedasticity and nonlinearity. This study will solely focus on the performance of GNIM and WGNIM without any pre-analysis of variable transformations, despite the fact that such transformations can improve the model fit.*

6.2.1 GAUSS-NEWTON ITERATIVE METHOD (GNIM)

The model is

$$y = \frac{e^{-\theta_1 x}}{(\theta_2 + \theta_3 x)} + \epsilon = f(x, \vec{\theta}) + \vec{\epsilon} \tag{6.1}$$

where $f(x, \vec{\theta})$ is the **expectation function** for the above 6.1 nonlinear regression model and the errors are uncorrelated and have normal distribution with mean of zero [15]. The sample model is given by

$$y_i = f(x_i, \vec{\theta}) + \vec{\epsilon}_i, \forall i = 1, 2, ..., 54, \tag{6.2}$$

where

$$\vec{\theta} = (\theta_1,\ \theta_2,\ \theta_3)',$$
$$\text{and } f(x_i, \vec{\theta}) = \frac{e^{-\theta_1 x_i}}{(\theta_2 + \theta_3 x_i)}, \forall i = 1, 2, ..., 54. \tag{6.3}$$

Equation (6.2) can be written as,

$$\epsilon_i = y_i - f(x_i, \vec{\theta}).$$

Then the sum of squares of the residuals is,

$$S(\vec{\theta}) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} [y_i - f(x_i, \vec{\theta})]^2.$$

Equation (4.10) can be obtained after converting the given nonlinear regression model to linear regression model using linearization method which has already been discussed in the previous section 4.2.

According to equation (4.10) for this problem $n = 54$, $p = 3$ (p is the number of parameters) and the Jacobian matrix consists of the following components.

$$\begin{aligned} z_{i1} &= \frac{\partial f(x_i, \vec{\theta})}{\partial \theta_1} = \frac{-x_i e^{-\theta_1 x_i}}{\theta_2 + \theta_3 x_i}, \\ z_{i2} &= \frac{\partial f(x_i, \vec{\theta})}{\partial \theta_2} = \frac{-e^{-\theta_1 x_i}}{(\theta_2 + \theta_3 x_i)^2}, \\ z_{i3} &= \frac{\partial f(x_i, \vec{\theta})}{\partial \theta_3} = \frac{-x_i e^{-\theta_1 x_i}}{(\theta_2 + \theta_3 x_i)^2}. \end{aligned} \tag{6.4}$$

The Jacobian matrix is given by

$$Z = (z_{ij}), \forall i = 1, 2, \ldots 54; \forall j = 1, 2, 3, \tag{6.5}$$

i.e.,

$$Z = \begin{bmatrix} \frac{\partial f(x_1,\vec{\theta})}{\partial \theta_1} & \frac{\partial f(x_1,\vec{\theta})}{\partial \theta_2} & \frac{\partial f(x_1,\vec{\theta})}{\partial \theta_3} \\ \frac{\partial f(x_2,\vec{\theta})}{\partial \theta_1} & \frac{\partial f(x_2,\vec{\theta})}{\partial \theta_2} & \frac{\partial f(x_2,\vec{\theta})}{\partial \theta_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial f(x_n,\vec{\theta})}{\partial \theta_1} & \frac{\partial f(x_n,\vec{\theta})}{\partial \theta_2} & \frac{\partial f(x_n,\vec{\theta})}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} \frac{-x_1 e^{-\theta_1 x_1}}{\theta_2+\theta_3 x_1} & \frac{-e^{-\theta_1 x_1}}{(\theta_2+\theta_3 x_1)^2} & \frac{-x_1 e^{-\theta_1 x_1}}{(\theta_2+\theta_3 x_1)^2} \\ \frac{-x_2 e^{-\theta_1 x_2}}{\theta_2+\theta_3 x_2} & \frac{-e^{-\theta_1 x_2}}{(\theta_2+\theta_3 x_2)^2} & \frac{-x_2 e^{-\theta_1 x_2}}{(\theta_2+\theta_3 x_2)^2} \\ \vdots & \vdots & \vdots \\ \frac{-x_n e^{-\theta_1 x_n}}{\theta_2+\theta_3 x_n} & \frac{-e^{-\theta_1 x_n}}{(\theta_2+\theta_3 x_n)^2} & \frac{-x_n e^{-\theta_1 x_n}}{(\theta_2+\theta_3 x_n)^2} \end{bmatrix}.$$

The residual vector is given by

$$\vec{y} - \vec{f} = \begin{bmatrix} y_1 - f_1 \\ y_2 - f_2 \\ \vdots \\ y_n - f_n \end{bmatrix} = \begin{bmatrix} y_1 - f(x_1, \vec{\theta}) \\ y_2 - f(x_2, \vec{\theta}) \\ \vdots \\ y_n - f(x_n, \vec{\theta}) \end{bmatrix} = \begin{bmatrix} y_1 - \frac{e^{-\theta_1 x_1}}{(\theta_2+\theta_3 x_1)} \\ y_2 - \frac{e^{-\theta_1 x_2}}{(\theta_2+\theta_3 x_2)} \\ \vdots \\ y_n - \frac{e^{-\theta_1 x_3}}{(\theta_2+\theta_3 x_3)} \end{bmatrix}. \tag{6.6}$$

The required Gauss-Newton recursion formula is as follows:

$$\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + ((Z^{(k)})' Z^{(k)})^{-1} (Z^{(k)})' (\vec{y} - \vec{f}^{(k)}), \forall k = 1, 2, ... \tag{6.7}$$

Fitting nonlinear models with iterative approaches requires initial values. When initial values are not suitable, the estimated fit parameters may converge to a local maximum or minimum instead of the global minimum or maximum. While some models are nearly unaffected by initial settings, others are hypersensitive.

For this particular problem consider the following starting values:

$$\theta_1^{(0)} = 0.1, \quad \theta_2^{(0)} = 0.01, \quad \theta_3^{(0)} = 0.02$$

The computation process of GNIM describes in the following steps:

Step 1: Set an initial approximation for $k = 0$,

$$\vec{\theta}^{(k)} = \vec{\theta}^{(0)} = (\theta_1^{(0)} = 0.1,\ \theta_2^{(0)} = 0.01,\ \theta_3^{(0)} = 0.02),$$

maximum iteration =1000, tolerance = 0.000001.

Step 2: Estimates the residual function $(\vec{y} - \vec{f}^{(k)})$ from equation (6.6) for each iteration $k = 1, 2, 3, \ldots$

Step 3: Simultaneously compute z_{i1}, z_{i2} and z_{i3}, $\forall i = 1, 2, \ldots, 54$ from equation (6.4).

Step 4: Calculate $\hat{\vec{\beta}}^{(k)}$ from equation (4.11).

Step 5: Update $\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + \hat{\vec{\beta}}^{(k)}$.

Step 6: If $\frac{\hat{\theta}_j^{(k+1)} - \hat{\theta}_j^{(k)}}{\hat{\theta}_j^{(k)}} < 0.000001$, then stop.

Step 7: Otherwise, set $k = k + 1$.

Then, go to the **step 2**.

Repeat steps 2 to 7 until convergence.

The nonlinear fit produced the following outcomes:

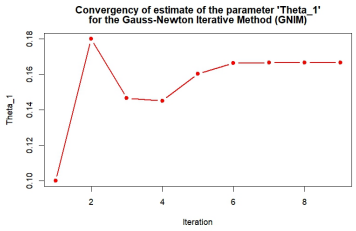
Table 6.2: Parameter estimate values of θ_1, θ_2, and θ_3, convergence values, residual values for ultrasonic calibration data at $\vec{\theta}^{(0)} = (\theta_1^{(0)} = 0.1, \theta_2^{(0)} = 0.01, \theta_3^{(0)} = 0.02)$.

Iteration	θ_1	θ_2	θ_3	Convergence metric	SS_{res}
1	0.1000000000	0.01000000	0.02000000	1.0000000000	14794.7901548
2	0.1799622604	0.00042163	0.00865227	0.7255999812	119693.8302051
3	0.1466505522	0.00147361	0.01217025	2.7164829605	16162.1819910
4	0.1450657284	0.00341316	0.01336677	1.4036933240	1392.4163047
5	0.1604124158	0.00490326	0.01248605	0.4764748822	519.5933359
6	0.1662118095	0.00515986	0.01216258	0.0625798088	513.0492894
7	0.1665600136	0.00516510	0.01215060	0.0021249250	513.0480312
8	0.1665759100	0.00516531	0.01215003	0.0000904151	513.0480294
9	0.1665766312	0.00516532	0.01215000	0.0000041060	513.0480294

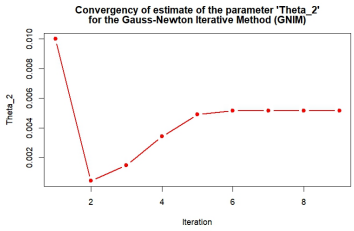
The values of the residual sum of squares (SS_{res}) for each iteration of the model fitting procedure are displayed in Table 6.2. A consistent decrease in the SS_{res} was noted

with each subsequent iteration. As shown in Table 6.2, the residual sum of squares at the beginning point is $S(\vec{\theta}^{(0)}) = 14794.7901548$, and the residual sum of squares at the final iteration is $S(\vec{\theta}^{(9)}) = 513.0480294$, which is a substantially smaller value than $S(\vec{\theta}^{(0)})$. The decrease in SS_{res} signifies that the model is gradually enhancing its association with the data, iteratively reducing the discrepancies between the predicted and observed values.

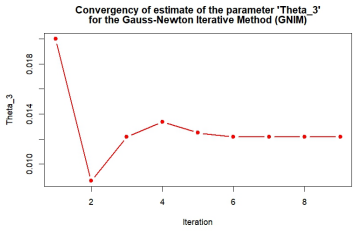
The provided figures depict the convergence of the parameters θ_1, θ_2, and θ_3 after nine iterations of implementing the Gauss-Newton Iterative Method (GNIM) on a nonlinear dataset for ultrasonic calibration. The x-axis of these images represents the iteration number, while the y-axis represents the parameter values. This allows for a graphical illustration of how each parameter gradually gets closer to its highest possible value. Figures 6.2a, 6.2b, and 6.2c display the path followed by θ_1, θ_2, and θ_3 during the iterations. As the iterations continue, the values of θ_1, θ_2, and θ_3 reach a stable state, indicating that the parameter value has reached convergence.



(a) The convergence path of the estimated coefficient θ_1 for GNIM.



(b) The convergence path of the estimated coefficient θ_2 for GNIM.



(c) The convergence path of the estimated coefficient θ_3 for GNIM.

Figure 6.2: Graphical illustration of the convergence of the estimated coefficients of θ_1, θ_2, and θ_3 for GNIM.

The efficacy of the GNIM in parameter estimation is illustrated through the convergence patterns that have been shown in the Figures 6.2a, 6.2b, and 6.2c. The parameters (θ_1, θ_2, and θ_3) of the nonlinear model (6.1) are optimally fitted, as evidenced by the uni-

form trajectory that conforms to a stable value. The convergence of all parameters (θ_1, θ_2, and θ_3) by the ninth iteration indicates that the convergence criterion has been satisfied, and no further iterations are required to produce significant changes.

After 9 iterations, for the Gauss-Newton iterative method, the estimated coefficients converged at $\theta_1 = 0.1665766312$, $\theta_2 = 0.00516532$, and $\theta_3 = 0.01215000$ with $S(\hat{\vec{\theta}}) = 513.0480294$. Consequently, the linearization process yielded the following fitted model:

$$\hat{y} = \frac{e^{\hat{\theta}_1 x}}{\hat{\theta}_2 + \hat{\theta}_3 x} = \frac{e^{0.1665766312x}}{0.00516532 + 0.01215000x} \tag{6.8}$$

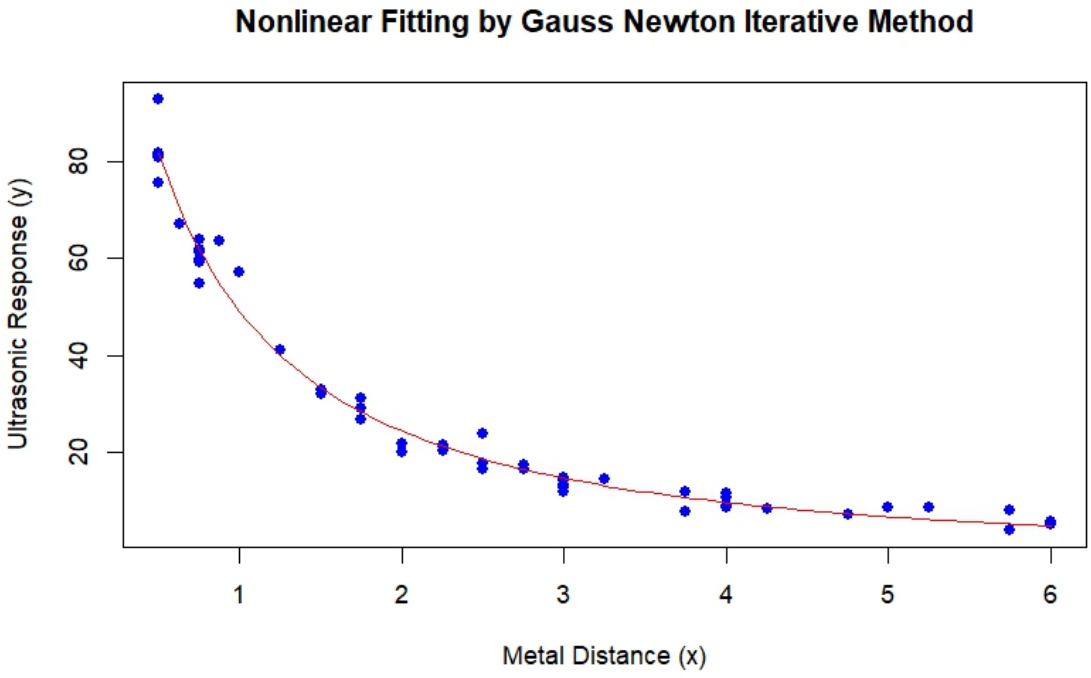


Figure 6.3: Graphical view of the nonlinear fitting process for ultrasonic calibration dataset using Gauss-Newton Iterative Method (GNIM).

The fitted model is shown in Figure 6.3. The plot demonstrates a satisfactory level of fit. Identifying any violations of the assumptions of the nonlinear model for this nonlinear fitting from Figure 6.3 presents a significant challenge. It is known that, a normal

distribution with a mean of zero and a constant variance is assumed to be the fundamental basis for regression models. But Figure 6.4 represents the violation of the constant variance assumptions.

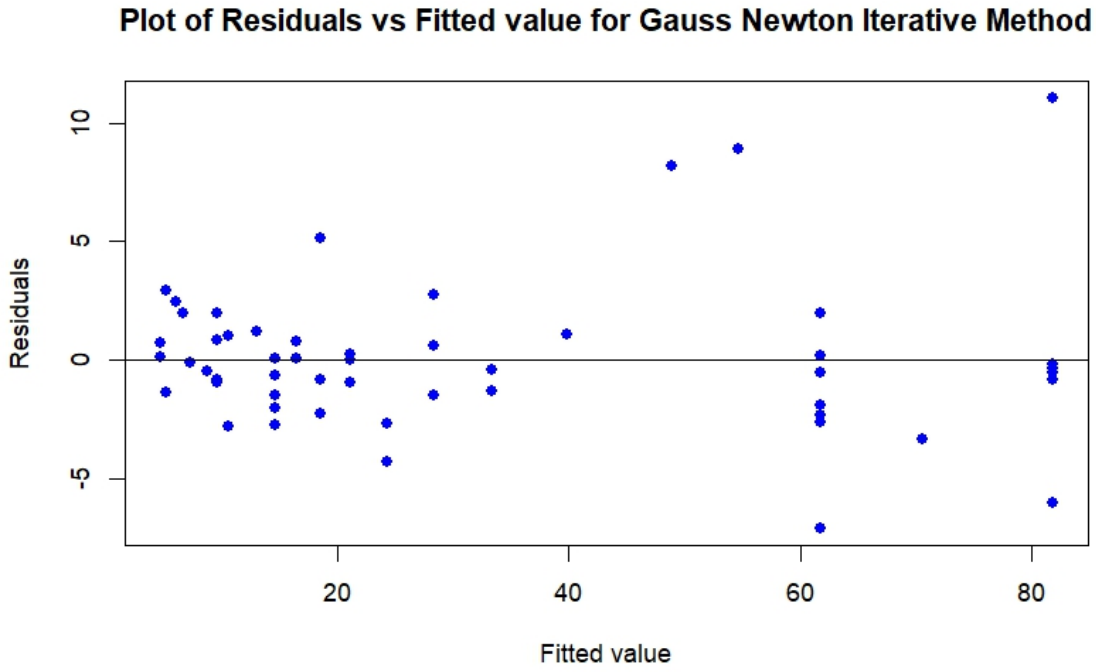


Figure 6.4: Graphical representation of residuals vs fitted values for Gauss-Newton Iterative Method (GNIM).

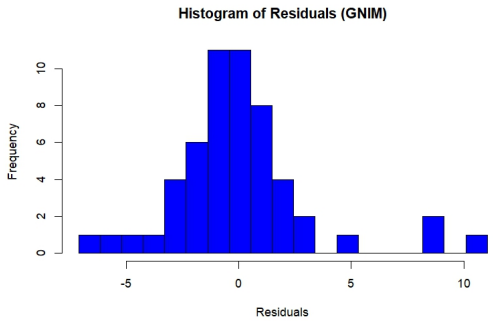


Figure 6.5: Graphical view of histogram of residuals for GNIM.

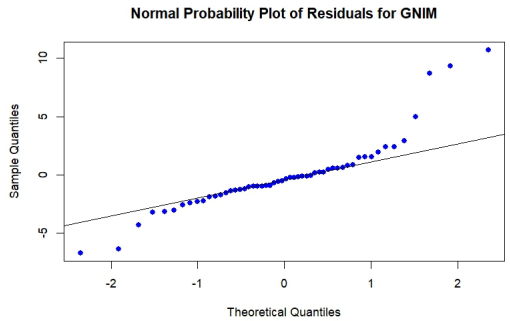


Figure 6.6: Graphical view of normal probability plot of residuals for GNIM.

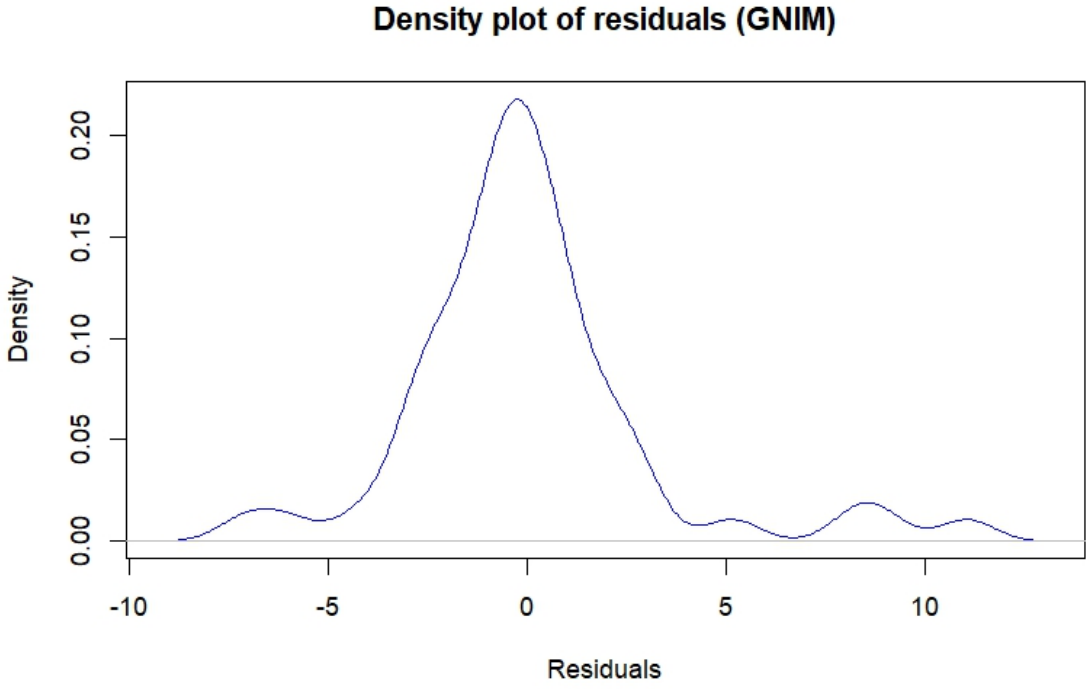


Figure 6.7: Graphical representation of density plot of residuals for GNIM.

After completing the model fitting process, an analysis was conducted on the residuals by examining their histogram and normal probability plot. The histogram plot in Figure 6.5 indicated a slight departure from normality since the distribution of residuals could not conform to the perfect shape of the expected bell-shaped curve. In addition, the normal probability plot in Figure 6.6 indicated that several points deviated from a straight line, providing more evidence that the residuals did not conform to an ideal normal distribution. The histogram diagram (6.5) can be further verified by the density plot diagram (6.7) to check whether the residuals are uniformly distributed or not. Since the density plot is almost bell-shaped according to the output, it assures that the residuals are not smoothly normally distributed, while a roughly bell-shaped plot indicates that the residuals are likely to follow a normal distribution.

Incorporating weights into the analysis is crucial for fixing these flaws and making

the model fit better. To improve the accuracy and reliability of the parameter estimations, weighted regression methods can be used to account for heteroscedasticity and other violations of the assumptions of the model. In the next subsection 6.2.2, weights will be introduced to mitigate this issue.

6.2.2 Weighting to Improve Fit

Conducting a weighted fit serves as an alternative method if the assumption of constant variance of the errors fails. When estimating the unknown parameters in the model, it is advisable to give less priority to less precise data and more importance to more precise measurements when using a weighted fit.

Weighted Gauss-Newton Iterative Method (WGNIM)

In this section, WGNIM will be applied to the model (6.1) with the starting values $\theta_1^{(0)} = 0.1$, $\theta_2^{(0)} = 0.01$, and $\theta_3^{(0)} = 0.02$, which are considered in the previous section 6.2.1 in the case of GNIM. Since Figure 6.4 indicates the violation of the constant variance assumption, in this section, the weights of each observation would be incorporated into the GNIM to resolve this issue.

To mitigate this inequality of variance problem, it is important to have knowledge of weights, w_i. After examining the data in Table 6.1, it is noticeable that there are several sets of x values that are replicated, that is, that have repeat points on x. The variance of the responses at those repeat points plays an important role in investigating how $\mathbf{Var}(y)$ changes with x.

Table 6.3: Evaluating weights for replicate predictor values in the WGNIM algorithm-Ultrasonic Calibration dataset.

Obs. i	Metal dis., x_i	Ult. res., y_i	$\bar{x}$	S_y^2	weight, w_i
1	0.5	92.90			0.09418060514
18	0.5	81.50			$\cdots$
28	0.5	75.80			$\cdots$
48	0.5	81.00	0.5	31.65466667	$\cdots$
51	0.5	81.70			$\cdots$
53	0.5	81.30			$\cdots$
10	0.75	63.80	0.75	8.182857143	0.09850234141
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Columns (4) and (5) of Table 6.3 show the average ($\bar{x}$) values for each cluster of the replicate values of x and the sample variance of the y in each corresponding cluster respectively. Plotting S_y^2 against the corresponding $\bar{x}$ implies that S_y^2 decreases approximately linearly with x. Now, after regressing S_y^2 against the $\bar{x}$, the least-square fit is given below:

$$\hat{S_y}^2 = 11.549606 - 1.863418\bar{x}. \tag{6.9}$$

The variance of the corresponding observation y_i can be estimated by substituting each predictor value x_i into equation (6.9). Reasonable estimates of the weights w_i will be the inverse of these fitted values. The final column of Table 6.3 displays these estimated weights.

Now from equation (6.4) calculate the elements z_{i1}, z_{i2}, and z_{i3} of the Jacobian matrix as well as from equation (6.6) evaluate the residual values and using this calculated jacobian matrix, residual and weighted values in Weighted Gauss-Newton recursion

formula 5.5 gives the following fitted model

$$\hat{y} = \frac{e^{\hat{\theta}_1 x}}{\hat{\theta}_2 + \hat{\theta}_3 x} = \frac{e^{0.13603758x}}{0.004697202 + 0.013336833x}. \tag{6.10}$$

The weighted fit results are shown in the Table 6.4. The computational algorithm of WGNIM follows the following steps:

Step 1: Set an initial approximation for $k = 0$,

$$\vec{\theta}^{(k)} = \vec{\theta}^{(0)} = (\theta_1^{(0)} = 0.1, \theta_2^{(0)} = 0.01, \theta_3^{(0)} = 0.02),$$

maximum iteration = 1000, tolerance = 0.000001.

Step 2: Arrange the dataset descending to ascending order and cluster the datasets with replications at each x_i values.

Step 3: Find out the mean values of predictors and sample variance of the corresponding response variables for each cluster.

Step 4: Regress the sample variances S_y^2 against the average values of $\bar{x}$, i.e.,

$$S_y^2 \sim \gamma_0 + \gamma_1 \bar{x} \tag{6.11}$$

where γ_0 and γ_1 are the intercept and slope of this regression model respectively.

Step 5: Substituting each x_i value into the equation (6.11) will give the estimate of the variance σ_i^2 of the corresponding observation y_i.

Step 6: Calculate $w_i = \frac{1}{\sigma_i^2}$.

Step 7: Estimates the residual function $(\vec{y} - \vec{f}^{(k)})$ from (6.6).

Step 8: Simultaneously compute $z_{i,1}$, $z_{i,2}$ and $z_{i,3}$, $\forall i = 1, 2, \ldots, 54$ from (6.4).

Step 9: Estimate $\hat{\vec{\beta}}^{(k)}$ using equation (5.4).

Step 10: Update $\hat{\vec{\theta}}^{(k+1)} = \hat{\vec{\theta}}^{(k)} + \hat{\vec{\beta}}^{(k)}$ using Weighted Gauss-Newton recursion formula (5.5).

Step 11: If $\frac{(\hat{\theta}_j^{(k+1)} - \hat{\theta}_j^{(k)})}{\hat{\theta}_j^{(k)}} < 0.000001$, then stop

Step 12: Otherwise, set $k = k + 1$.

go to the **step 7**.

Repeat Steps 7 to 12 until convergence.

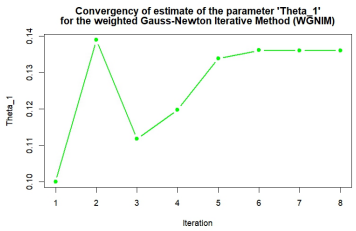
Table 6.4: Estimated coefficient values of θ_1, θ_2, and θ_3, convergence values, values of SS_{res} for the "Ultrasonic Calibration data" at $\vec{\theta}^{(0)} = (\theta_1^{(0)} = 0.1, \theta_2^{(0)} = 0.01, \theta_3^{(0)} = 0.02)$.

Iteration	θ_1	θ_2	θ_3	Convergence metric	SS_{res}
1	0.10000000	0.010000000	0.020000000	1.00000000000	1502.08058804
2	0.13890199	-0.000707836	0.011459040	1.10881170857	9138.42007790
3	0.11177751	0.000462726	0.014885091	1.55001422993	1185.29440401
4	0.11973696	0.002975907	0.014794506	5.49637732339	125.08227239
5	0.13385763	0.004552942	0.013483586	0.55925618232	70.90529543
6	0.13605905	0.004698578	0.013334030	0.03734152275	70.63140942
7	0.13603685	0.004697169	0.013336892	0.00024826878	70.63139000
8	0.13603758	0.004697202	0.013336833	0.00000797973	70.63139000

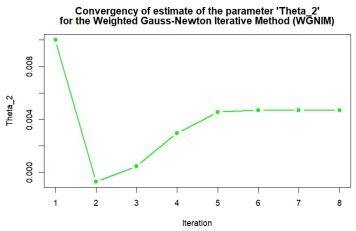
The values of the residual sum of squares (SS_{res}) for every iteration of the nonlinear model fitting technique, namely WGNIM, are displayed in Table 6.4. With every iteration, a constant reduction in SS_{res} was noted, mirroring the pattern found in the GNIM. At the beginning, the residual sum of squares was 1502.08058804, but by the last iteration, it had been reduced to 70.63139000. In addition to being a significant improvement over the GNIM, this final value is substantially lower than the first one. With each iteration, the model is becoming better at fitting the data by reducing the difference between the predicted and observed values, as seen by the consistent decline in SS_{res}.

As shown in Figures 6.8a, 6.8b, and 6.8c, the convergence of parameters θ_1, θ_2, and θ_3 for ultrasonic calibration was achieved through the implementation of the Weighted Gauss-Newton Iterative Method (WGNIM) on a nonlinear dataset. The provided figures illustrate the path that each parameter traveled throughout the iterative procedure.

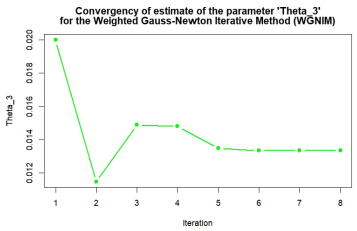
These graphs show convergence patterns, which prove that the WGNIM is effective for parameter estimation. It should be noted that the convergence pattern of the parameter is comparable to what is seen with the GNIM. In contrast, the WGNIM reaches convergence after just eight iterations, suggesting that it takes fewer iterations for the WGNIM to reach a fixed point than the GNIM.



(a) The convergence path of the estimated coefficient θ_1 for WGNIM.



(b) The convergence path of the estimated coefficient θ_2 for WGNIM.



(c) The convergence path of the estimated coefficient θ_3 for WGNIM.

Figure 6.8: Graphical illustration of the convergence path of the estimated coefficients of θ_1, θ_2, and θ_3 for WGNIM.

To evaluate the adequacy of the weighted fit, it is necessary to initially construct a graphical representation of the predicted line in relation to the original data. As indicated by Figure 6.9 of predicted values against data, a decent fit is present.

As seen in Figure 6.9, the model fitted the data very well after applying the Weighted Gauss-Newton Iterative Method (WGNIM). This is similar to the findings obtained with the Gauss-Newton Iterative Method (GNIM).

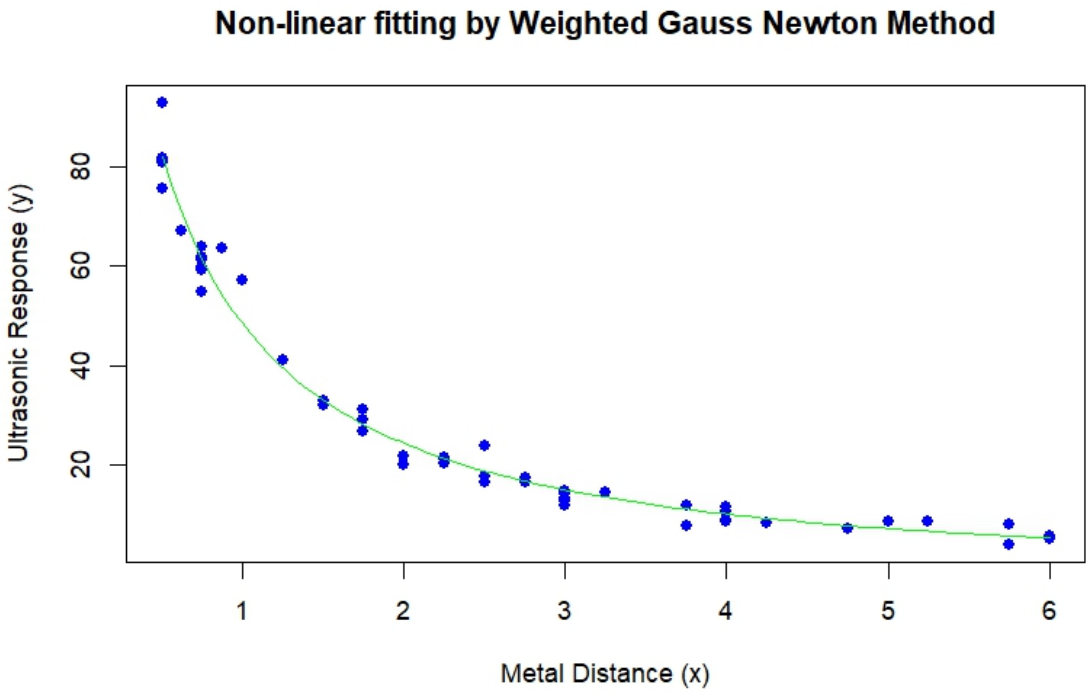


Figure 6.9: Graphical view of the nonlinear fitting process for ultrasonic calibration dataset using Weighted Gauss-Newton Iterative Method (WGNIM).

Following that, an evaluation of heteroscedasticity was performed, which unveiled the existence of this concern in the GNIM outcomes. Nevertheless, Figure 6.10 depicting the WGNIM implementation provided evidence that this approach effectively mitigated the issue of heteroscedasticity, as evidenced by the consistent variances observed throughout the data points, which means the weighted residuals are evenly distributed about the zero line, which is clearly shown in Figure 6.10.

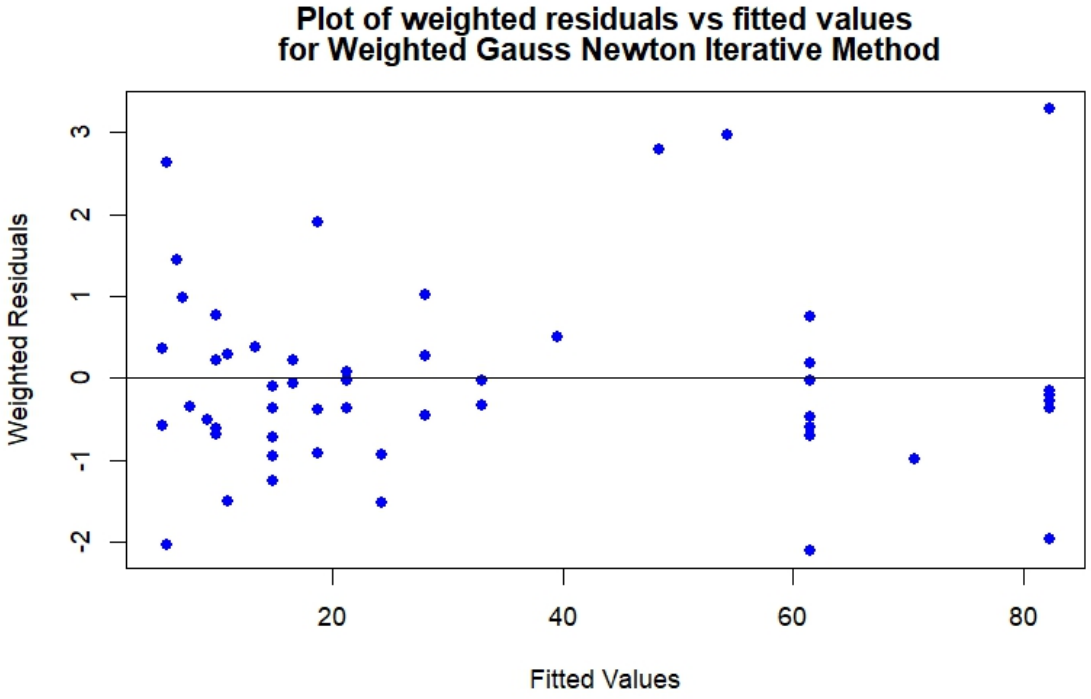


Figure 6.10: Graphical representation of weighted residuals vs fitted values for Weighted Gauss-Newton Iterative Method (WGNIM).

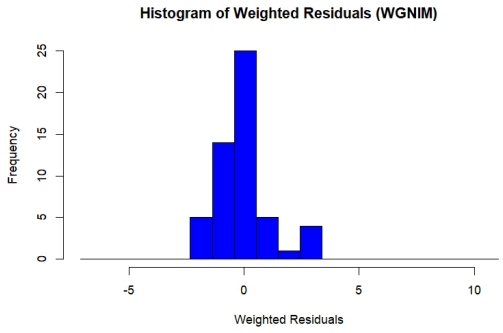


Figure 6.11: Histogram of weighted residuals for WGNIM.

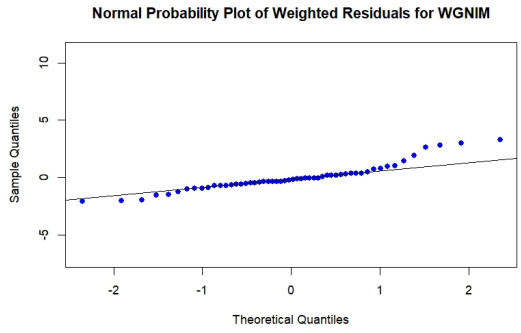


Figure 6.12: Normal probability plot of weighted residuals for WGNIM.

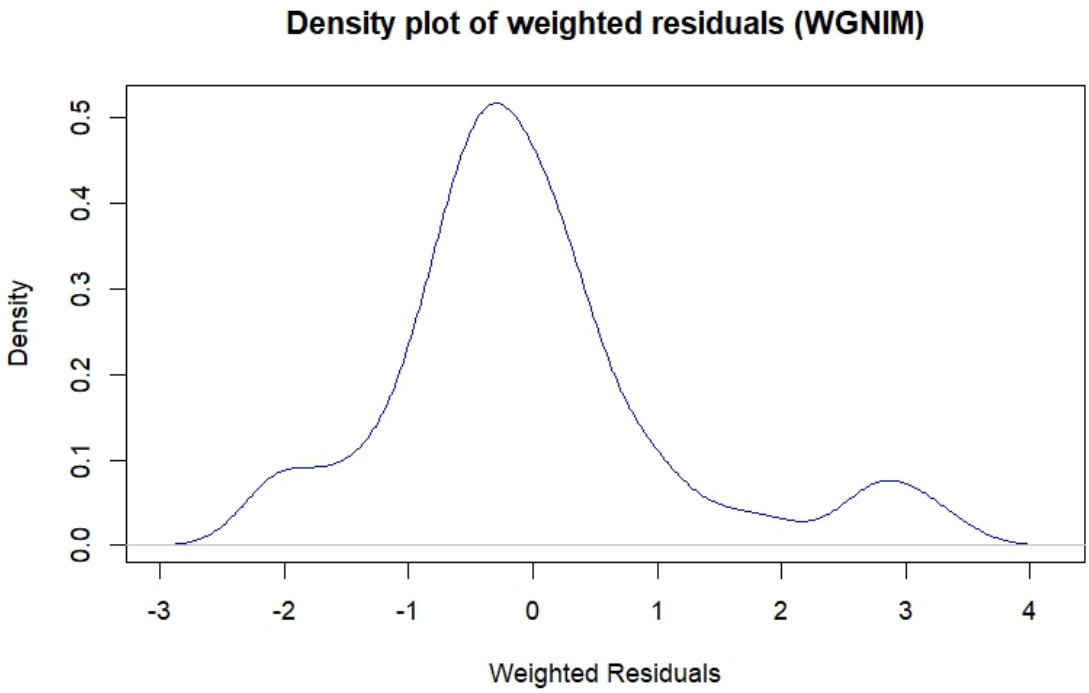


Figure 6.13: Graphical representation of density plot of weighted residuals for Weighted Gauss-Newton Iterative Method (WGNIM).

In addition, normal probability graphs and histograms were produced to illustrate the residuals for the WGNIM model. According to the diagrams, the WGNIM produced more favorable residual distributions than the GNIM, indicating that its results were superior. With regard to the histogram diagram (6.11), displayed better conformity to the anticipated bell-shaped curve. More precisely, the density plot of weighted residuals for WGNIM, which was modified by the square root of the weights, reveals a more normal distribution in comparison to GNIM, which indicates that there has been an improvement in the stabilization of variance, which is shown in Figure 6.13. In addition, the normal probability plot (6.12) exhibited a more noticeable alignment of data points along the straight line, suggesting a more favorable fit to the normal distribution.

With the implementation of the WGNIM, the model has successfully taken into

account heteroscedasticity. This technique makes use of adaptive weighting in order to account for the different variances that are present in the data. As a result, the coefficient estimations are more accurate, and the model fitting capability is improved. In spite of the fact that heteroscedasticity is still present in the data, the WGNIM makes improvements to account for it, which makes the findings more reliable.

6.2.3 Compare the Fits

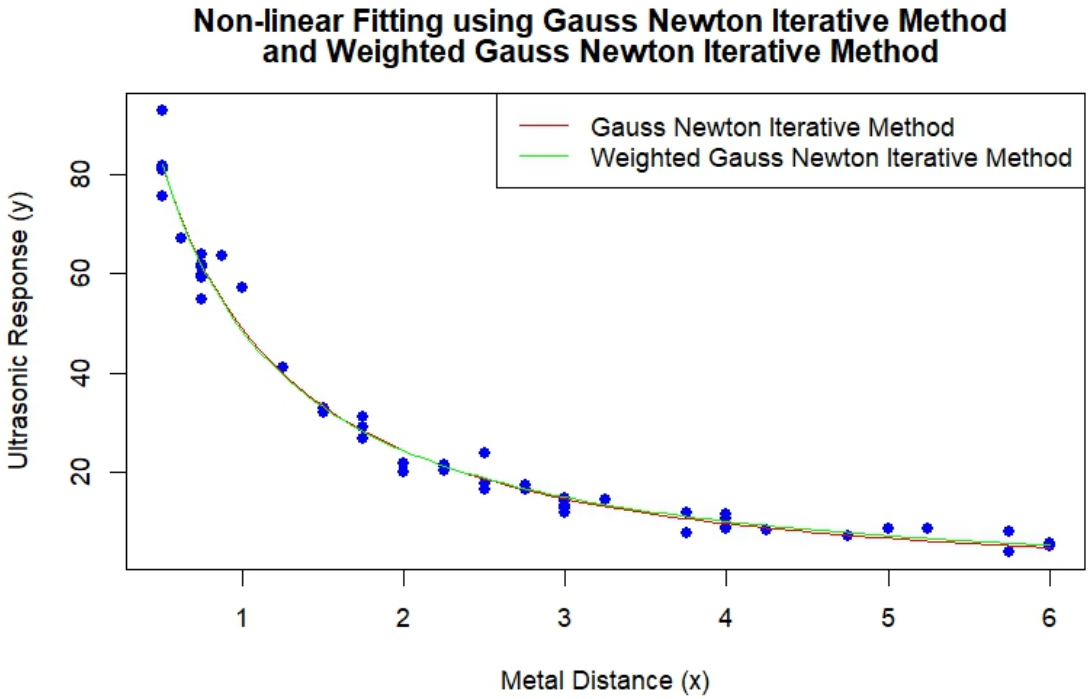
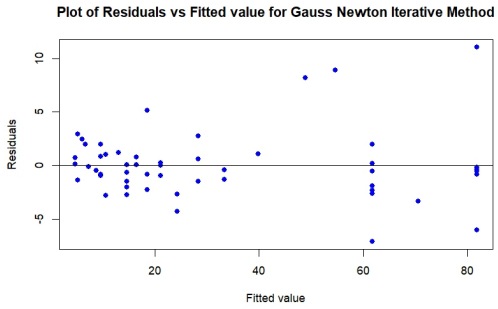
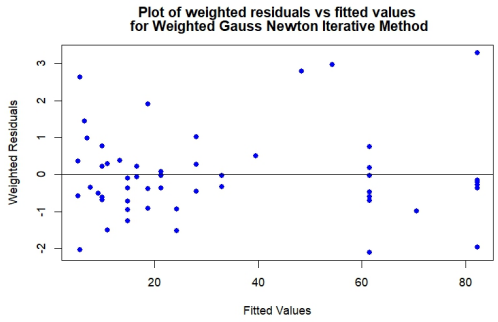


Figure 6.14: Graphical illustration of comparative nonlinear fitting process between GNIM and WGNIM for the ultrasonic calibration dataset.

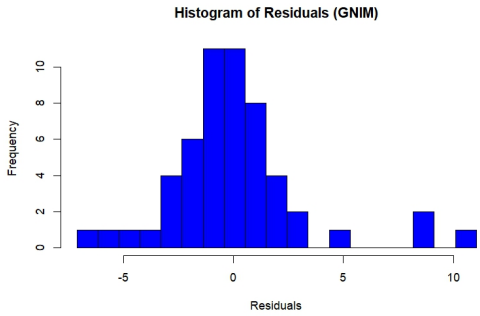


(a) Graphical view of residuals vs fitted values for ultrasonic calibration dataset using GNIM.

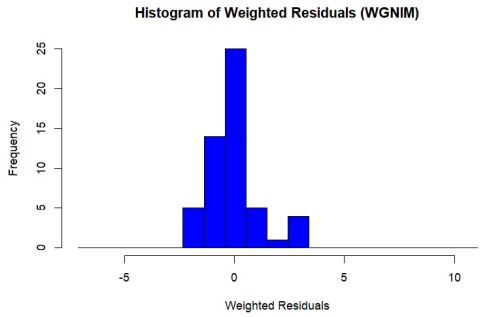


(b) Graphical view of weighted residuals vs weighted fitted values for ultrasonic calibration dataset using WGNIM.

Figure 6.15: Comparative graphical representation of residuals versus fitted values between GNIM and WGNIM.

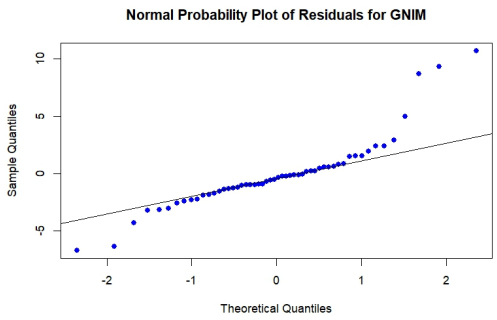


(a) Graphical view of histogram of residuals for GNIM.

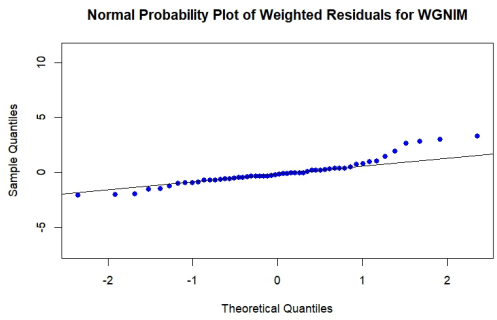


(b) Graphical view of histogram of weighted residuals for WGNIM.

Figure 6.16: Comparative graphical view of histogram of residuals between GNIM and WGNIM.

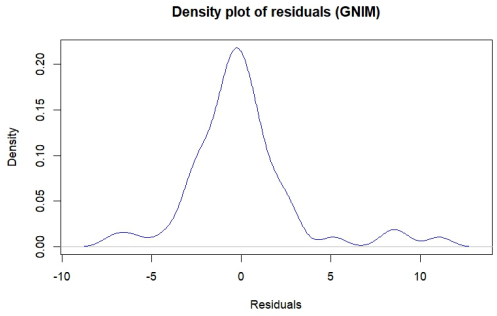


(a) Graphical view of normal probability plot of residuals for GNIM.

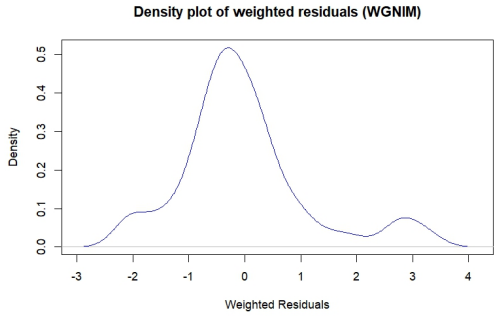


(b) Graphical view of normal probability plot of weighted residuals for WGNIM.

Figure 6.17: A visual comparison figure of the normal probability plot between GNIM and WGNIM.



(a) Graphical view of density of residuals for GNIM.



(b) Graphical view of density of weighted residuals for WGNIM.

Figure 6.18: Comparative visual illustration of density of residuals between GNIM and WGNIM.

To commence the comparison of fits, it is important to graphically represent the two sets of predicted values, expressed in the original units, on a shared plot alongside the raw data. The presented Figure 6.14 demonstrates that the two fits yield similar anticipated values. The utilization of weighted fits yields predicted values that exhibit a high degree

of proximity to the original fit. The regression estimations for WGNIM have exhibited minimal variation compared to GNIM which are shown in the Table 6.5. The WGNIM model is a modified version of the GNIM model that effectively addresses the prerequisites for fitting a nonlinear model. This provides us with assurance that the results and analysis derived from the WGNIM are well-founded and suitable.

Table 6.5: Comparison of different iterative methods for estimated parameter values of θ_1, θ_2, and θ_3 as well as MS_{res}.

Method	β_1	β_2	β_3	MS_{res}
GNIM	0.1665766312	0.00516532	0.01215000	10.05976528
WGNIM	0.13603758	0.004697202	0.013336833	1.38492922
Levenberg-Marquardt (with weight) (using nls function)	0.1360376	0.0046972	0.0133368	1.38492922
Levenberg-Marquardt (without weight) (using nls function)	0.166577	0.005165	0.012150	10.05976528

Table 6.5 summarizes that, compared to the unweighted methods, which have substantially higher MS_{res} values (10.05976528), the weighted methods have significantly lower MS_{res} values (1.38492922). A lower MS_{res} value suggests that the weighted algorithms offer a more satisfactory fit to the data.

Table 6.6: Execution time for each iteration in case of Gauss-Newton Iterative Method (GNIM).

Iteration	θ_1	θ_2	θ_3	Time	Cumulative Time
1	0.1000000000	0.01000000	0.02000000	0.01273393630981	0.01273393631
2	0.1799622604	0.00042163	0.00865227	0.00374102592468	0.01647496223
3	0.1466505522	0.00147361	0.01217025	0.00012207031250	0.01659703255
4	0.1450657284	0.00341316	0.01336677	0.00010013580322	0.01669716835
5	0.1604124158	0.00490326	0.01248605	0.00009202957153	0.01678919792
6	0.1662118095	0.00515986	0.01216258	0.00008988380432	0.01687908173
7	0.1665600136	0.00516510	0.01215060	0.00008702278137	0.01696610451
8	0.1665759100	0.00516531	0.01215003	0.00008296966553	0.01704907417
9	0.1665766312	0.00516532	0.01215000	0.00008106231689	0.01713013649

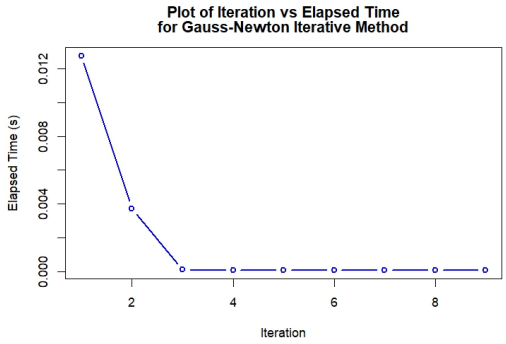


Figure 6.19: Plot of elapsed time for each iteration for GNIM.

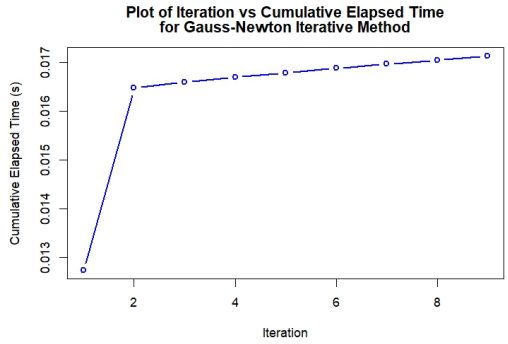


Figure 6.20: Plot of cumulative elapsed time for each iteration for GNIM.

Table 6.7: Execution time for each iteration in case of Weighted Gauss-Newton Iterative Method (WGNIM).

Iteration	θ_1	θ_2	θ_3	Time	Cumulative Time
1	0.10000000	0.010000000	0.020000000	0.0241878032684	0.02418780327
2	0.13890199	-0.000707836	0.011459040	0.0044209957123	0.02860879898
3	0.11177751	0.000462726	0.014885091	0.0003311634064	0.02893996239
4	0.11973696	0.002975907	0.014794506	0.0004050731659	0.02934503555
5	0.13385763	0.004552942	0.013483586	0.0003199577332	0.02966499329
6	0.13605905	0.004698578	0.013334030	0.0003671646118	0.03003215790
7	0.13603685	0.004697169	0.013336892	0.0004439353943	0.03047609329
8	0.13603758	0.004697202	0.013336833	0.0003080368042	0.03078413010

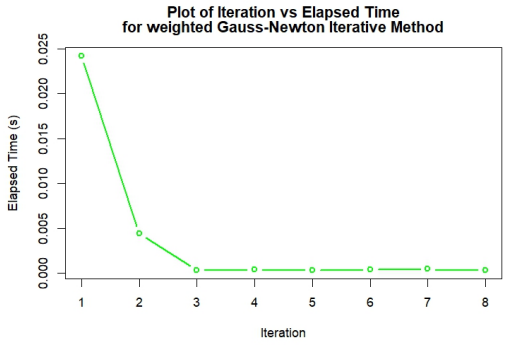


Figure 6.21: Graphical view of elapsed time for each iteration for Weighted Gauss-Newton Iterative Method (WGNIM).

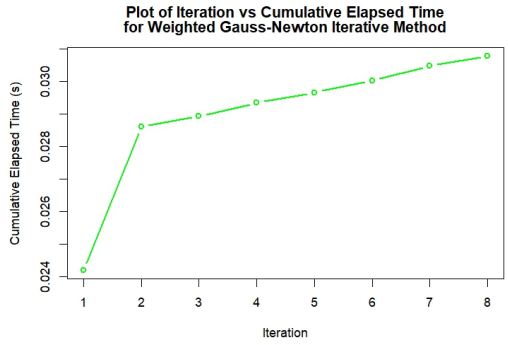


Figure 6.22: Graphical view of cumulative elapsed time for each iteration for weighted Gauss-Newton Iterative Method (WGNIM).

Table 6.8: Comparison of consecutive execution time and cumulative elapsed time between Gauss-Newton Iterative Method (GNIM) and Weighted Gauss-Newton Iterative Method (WGNIM).

Ite.No.	GNIM		WGNIM	
	Time	Cumulative Time	Time	Cumulative Time
1	0.01273393630981	0.01273393631	0.0241878032684	0.02418780327
2	0.00374102592468	0.01647496223	0.0044209957123	0.02860879898
3	0.00012207031250	0.01659703255	0.0003311634064	0.02893996239
4	0.00010013580322	0.01669716835	0.0004050731659	0.02934503555
5	0.00009202957153	0.01678919792	0.0003199577332	0.02966499329
6	0.00008988380432	0.01687908173	0.0003671646118	0.03003215790
7	0.00008702278137	0.01696610451	0.0004439353943	0.03047609329
8	0.00008296966553	0.01704907417	0.0003080368042	0.03078413010
9	0.00008106231689	0.01713013649	-	-

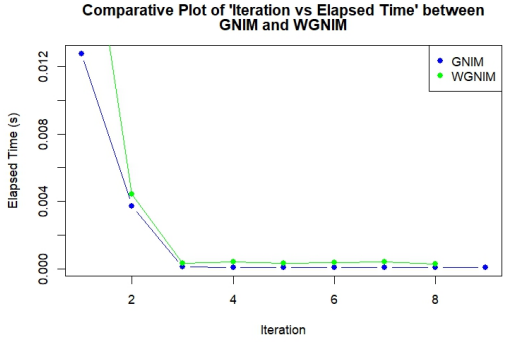


Figure 6.23: Comparative elapsed time for each iteration between GNIM and WGNIM.

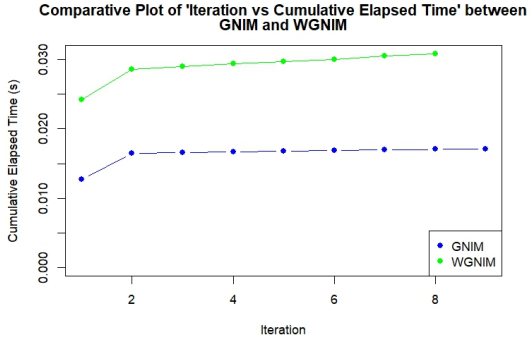


Figure 6.24: Comparative cumulative execution time for each iteration between GNIM and WGNIM.

The execution time comparison between the Weighted Gauss-Newton Iterative Method (WGNIM) and the Gauss-Newton Iterative Method (GNIM) for each iteration unveiled significant disparities. The execution duration of the WGNIM was significantly longer than

that of the GNIM, owing to the incorporation of weights into its algorithm.

More specifically, the WGNIM exhibited an extended execution time for every iteration, which can be attributed to the supplementary computational effort required to compute and implement the weights within the algorithm. In addition, the overall duration of the procedure was found to be longer for the WGNIM in comparison to the GNIM.

The observed variation in execution time underscores the trade off between the enhanced performance and the reduced computational efficiency that the WGNIM implements to account for heteroscedasticity. Although the execution time of the WGNIM may be longer, in certain applications, its capability to manage heteroscedasticity and generate more dependable parameter estimates may be sufficient to justify this additional computational expense.

6.2.4 Analysis of Statistical Inferences for Nonlinear Regression Parameter

Confidence Interval Estimation for GNIM

Estimated Variance and Covariance

Inferences about nonlinear regression parameters require an estimate the variance σ^2 of the error term. Following the convergence of the estimating technique to a final vector of parameter estimates $\hat{\theta}$, it is possible to derive an estimated variance σ^2 of the error by utilizing the residual mean square. This estimation is identical to that of linear regression.

$$\begin{aligned}\hat{\sigma}^2 = MS_{res} &= \frac{SS_{res}}{n-p} \\ &= \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-p} \\ &= \frac{\sum_{i=1}^{n}[\vec{Y_i} - f_i(X_i, \hat{\vec{\theta}})]^2}{n-p} \\ &= \frac{S(\hat{\vec{\theta}})}{n-p},\end{aligned} \tag{6.12}$$

where p is the number of parameters, and the vector, denoted by $\hat{\vec{\theta}}$, contains the final estimates of the parameters. Since MS_{res} is not an unbiased estimator of σ^2 in the case of nonlinear regression, with a large sample size, the bias would be quite negligible [14].

The following theorem serves as the foundation for drawing inferences for regression models in situations when the errors are independent and normally distributed, and the sample size is of a size that is considered to be reasonably large:

Large Sample theory

Firstly, when the error terms ϵ_i are considered to be independent and normally distributed with mean 0 and variance σ^2, which can be written as $\epsilon_i \sim N(0, \sigma^2)$ and the sample size n is assumed to be reasonably large, the sampling distribution of $\hat{\vec{\theta}}$ is approximately normal [14], i.e.,

$$E(\hat{\vec{\theta}}) \simeq \theta. \tag{6.13}$$

Secondly, the approximate variance-covariance matrix of the regression coefficients can be computed using the following formula [14]:

$$\mathbf{Var}(\hat{\vec{\theta}}) = MS_{res}(Z'Z)^{-1}, \tag{6.14}$$

where Z is the matrix of partial derivatives of the model function with respect to the parameters defined previously evaluated at the final iteration of the least-squares estimate $\hat{\vec{\theta}}$.

Remark 6.2. *It should be noted that the form of the estimated approximation variance-covariance matrix* $\mathbf{Var}(\hat{\vec{\theta}})$ *is identical to that of the linear regression matrix, with* Z *serving as the* X *matrix.*

In this application, the residual sum of squares at the final iteration is $S(\hat{\vec{\theta}}) = 513.0480294$. So the estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{S(\hat{\vec{\theta}})}{n-p} = \frac{513.0480294}{54-3} = 10.05976528235$$

The covariance matrix of the estimated coefficient vector $\hat{\vec{\theta}}$ for the given exponential model in this application is

$$\begin{aligned}\mathbf{Var}(\hat{\vec{\theta}}) &= MS_{res}(Z'Z)^{-1} \\ &= 10.05976528235 \begin{bmatrix} 1.45843 \times 10^{-04} & 2.14144 \times 10^{-06} & -5.47605 \times 10^{-06} \\ 2.14144 \times 10^{-06} & 4.41207 \times 10^{-08} & -9.75029 \times 10^{-08} \\ -5.47605 \times 10^{-06} & -9.75029 \times 10^{-08} & 2.32828 \times 10^{-07} \end{bmatrix}.\end{aligned}$$

The main diagonal elements of this matrix are approximate variances of the estimates of the regression coefficients. Therefore, approximate standard errors of the estimated coefficients are given as follows:

$$\begin{aligned} SE(\hat{\theta}_1) = \sqrt{\mathbf{Var}(\hat{\theta}_1)} &= \sqrt{10.05976528235(1.45843 \times 10^{-04})} \\ &= 0.03830334643 \end{aligned}$$

$$\begin{aligned} SE(\hat{\theta}_2) = \sqrt{\mathbf{Var}(\hat{\theta}_2)} &= \sqrt{10.05976528235(4.41207 \times 10^{-08})} \\ &= 0.0006662161 \end{aligned}$$

$$\begin{aligned} SE(\hat{\theta}_3) = \sqrt{\mathbf{Var}(\hat{\theta}_3)} &= \sqrt{10.05976528235(2.32828 \times 10^{-07})} \\ &= 0.00153042315 \end{aligned}$$

Interval Estimation of Regression Parameters

Based on large-sample theory, given a large sample size and normally distributed error terms, the following approximate result is valid [14].

$$\frac{\hat{\theta}_j - \theta_j}{SE(\hat{\theta}_j)} \sim t(n-p), \forall j = 1, 2, ..., p. \tag{6.15}$$

where j stands for a particular element of the parameter vector θ and p represents the total number of model parameters. The above statement asserts that, the standardize estimate $\frac{\hat{\theta}_j - \theta_j}{SE(\hat{\theta}_j)}$ for each parameter θ_j approximately follows a t-distribution with $n-p$ degrees of freedom, where n is the sample size.

For every parameter θ_j, this result is utilized to generate confidence intervals and do hypothesis testing. Therefore, a confidence interval for every θ_j with a confidence level of $100(1-\alpha)\%$ can be created as follows [14]:

$$\hat{\theta}_j \pm t\left(1 - \frac{\alpha}{2}, n-p\right) \cdot SE(\hat{\theta}_j), \tag{6.16}$$

where the critical value, denoted as $t(1-\frac{\alpha}{2}, n-p)$, is obtained from t-distribution with $n-p$ degrees of freedom.

For our ultrasonic calibration example, it is desired to estimate θ_1, θ_2, and θ_3 with a 95% confidence interval.

Assume that,

Confidence level, CL = 95%,

Level of significance, $\alpha = 5\% = 0.05$,

Confidence coefficient, $1 - \frac{\alpha}{2} = 1 - \frac{0.05}{2} = 0.975$,

The sample size, $n = 54$,

The total number of parameter, $p = 3$,

Therefore, the required critical value is

$$t\left(1-\frac{\alpha}{2}, n-p\right) = t(0.975, 51)$$
$$= 2.007584$$

Confidence interval for θ_1:

The value of estimated coefficient θ_1 is $\hat{\theta}_1 = 0.1665766638$ and standard error of the coefficient estimates of θ_1 is $SE(\hat{\theta}_1) = 0.00146714635$.

Then a 95% confidence interval for θ_1 is as follows,

$$\hat{\theta}_1 - t_{0.025,51} \cdot SE(\hat{\theta}_1) \leq \theta_1 \leq \hat{\theta}_1 + t_{0.025,51} \cdot SE(\hat{\theta}_1)$$

$$0.1665767 - 2.007584 \times 0.03830334643 \leq \theta_1 \leq 0.1665767 + 2.007584 \times 0.03830334643$$

$$0.08967951 \leq \theta_1 \leq 0.24347389$$

Interpretation of confidence interval for θ_1:

Therefore, the interval from 0.08967951 to 0.24347389 forms a 95% confidence interval for the estimated coefficient θ_1. In other words, the interval from 0.08967951 to 0.24347389 gives the most believable value for the parameter estimate θ_1.

Confidence interval for θ_2:

The value of estimated coefficient θ_2 is $\hat{\theta}_2 = 0.00516533$ and standard error of the coefficient estimates of θ_2 is $SE(\hat{\theta}_2) = 0.0006662161$.

Then a 95% confidence interval for θ_2 is as follows,

$$\hat{\theta}_2 - t_{0.025,51} \cdot SE(\hat{\theta}_2) \leq \theta_2 \leq \hat{\theta}_2 + t_{0.025,51} \cdot SE(\hat{\theta}_2)$$

$$0.00516533 - 2.007584 \times 0.0006662161 \leq \theta_2 \leq 0.00516533 - 2.007584 \times 0.0006662161$$

$$0.00382785 \leq \theta_2 \leq 0.00650281$$

Interpretation of confidence interval for θ_2:

Therefore, the interval from 0.00382785 to 0.00650281 forms a 95% confidence interval for the estimated coefficient θ_2. In other words, the interval from 0.00382785 to 0.00650281 gives the most believable value for the parameter estimate θ_2.

Confidence interval for θ_3:

The value of estimated coefficient θ_3 is $\hat{\theta}_3 = 0.01215001$ and standard error of the coefficient estimates of θ_3 is $SE(\hat{\theta}_3) = 0.001530423$.

Then a 95% confidence interval for θ_3 is as follows,

$$\hat{\theta}_3 - t_{0.025,51} \cdot SE(\hat{\theta}_3) \leq \theta_3 \leq \hat{\theta}_3 + t_{0.025,51} \cdot SE(\hat{\theta}_3)$$

$$0.01215001 - 2.007584 \times 0.001530423 \leq \theta_3 \leq 0.01215001 + 2.007584 \times 0.001530423$$

$$0.00907756 \leq \theta_3 \leq 0.01522246$$

Interpretation of confidence interval for θ_3:

Therefore, the interval from 0.00907756 to 0.01522246 forms a 95% confidence interval for the estimated coefficient θ_3. In other words, the interval from 0.00907756 to 0.01522246 gives the most believable value for the parameter estimate θ_3.

Confidence Interval Estimation for WGNIM

In case of WGNIM, the approximate variance-covariance matrix of the regression coefficients is estimated by:

$$\mathbf{Var}(\hat{\vec{\theta}}) = MS_{res}(Z'WZ)^{-1} \tag{6.17}$$

where Z is the matrix of partial derivatives of the expectation function of the given model with respect to the given parameters defined previously, evaluated at the final-iteration least-squares estimate $\hat{\vec{\theta}}$.

In case of WGNIM, weighted mean square error is

$$\begin{aligned}\hat{\sigma}^2 = MS_{res} &= \frac{\sum_{i=1}^{n} w_i(Y_i - \hat{Y}_i)^2}{n-p} \\ &= \frac{\sum_{i=1}^{n} w_i[\vec{Y_i} - f_i(X_i, \hat{\vec{\theta}})]^2}{n-p} \\ &= \frac{\sum_{i=1}^{n} w_i e_i^2}{n-p} \\ &= \frac{S(\hat{\vec{\theta}})}{n-p}. \end{aligned} \tag{6.18}$$

In this application, the required weighted residual sum of squares at the final iteration is $S(\hat{\vec{\theta}}) = 70.63139000$. So the estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{S(\hat{\vec{\theta}})}{n-p} = \frac{70.63139000}{54-3} = 1.384929216.$$

The covariance matrix of the estimated coefficient vector $\hat{\vec{\theta}}$ for the given model in this application is given below:

$$\begin{aligned}\mathbf{Var}(\hat{\vec{\theta}}) &= MS_{res}(Z'WZ)^{-1} \\ &= 1.384929216 \begin{bmatrix} 3.01590 \times 10^{-04} & 7.44436 \times 10^{-06} & -1.63246 \times 10^{-05} \\ 7.44436 \times 10^{-06} & 2.76268 \times 10^{-07} & -5.14424 \times 10^{-07} \\ -1.63246 \times 10^{-05} & -5.14424 \times 10^{-07} & 1.05687 \times 10^{-06} \end{bmatrix}.\end{aligned}$$

The main diagonal elements of this matrix are approximate variances of the estimates of the regression coefficients. Therefore, approximate standard errors on the coefficients are

$$\begin{aligned} SE(\hat{\theta}_1) = \sqrt{\mathbf{Var}(\hat{\theta}_1)} &= \sqrt{1.384929216(3.01590 \times 10^{-04})} \\ &= 0.020437241 \end{aligned}$$

$$SE(\hat{\theta}_2) = \sqrt{\mathbf{Var}(\hat{\theta}_2)} = \sqrt{1.384929216(2.76268 \times 10^{-07})}$$
$$= 0.000618557$$

$$SE(\hat{\theta}_3) = \sqrt{\mathbf{Var}(\hat{\theta}_3)} = \sqrt{1.384929216(1.05687 \times 10^{-06})}$$
$$= 0.001209831$$

Confidence interval for θ_1:

The value of estimated coefficient θ_1 is $\hat{\theta}_1 = 0.13603758$ and standard error of the coefficient estimates of θ_1 is $SE(\hat{\theta}_1) = 0.020437241$.

Then a 95% confidence interval for θ_1 is as follows,

$$\hat{\theta}_1 - t_{0.025,51} \cdot SE(\hat{\theta}_1) \leq \theta_1 \leq \hat{\theta}_1 + t_{0.025,51} \cdot SE(\hat{\theta}_1)$$
$$0.13603758 - 2.007584 \times 0.020437241 \leq \theta_1 \leq 0.13603758 + 2.007584 \times 0.020437241$$
$$0.0950086 \leq \theta_1 \leq 0.177066574$$

Interpretation of confidence interval for θ_1:

Therefore, the interval from 0.0950086 to 0.177066574 forms a 95% confidence interval for the estimated coefficient θ_1. In other words, the interval from 0.0950086 to 0.177066574 gives the most believable value for the parameter estimate θ_1.

Confidence interval for θ_2:

The value of estimated coefficient θ_2 is $\hat{\theta}_2 = 0.004697202$ and standard error of the coefficient estimates of θ_2 is $SE(\hat{\theta}_2) = 0.000618557$.

Then a 95% confidence interval for θ_2 is as follows,

$$\hat{\theta}_2 - t_{0.025,51} \cdot SE(\hat{\theta}_2) \leq \theta_2 \leq \hat{\theta}_2 + t_{0.025,51} \cdot SE(\hat{\theta}_2)$$
$$0.004697202 - 2.007584 \times 0.000618557 \leq \theta_2 \leq 0.004697202 + 2.007584 \times 0.000618557$$
$$0.00345540 \leq \theta_2 \leq 0.00593901$$

Interpretation of confidence interval for θ_2:

Therefore, the interval from 0.00345540 to 0.00593901 forms a 95% confidence interval for the estimated coefficient θ_2. In other words, the interval from 0.00345540 to 0.00593901 gives the most believable value for the parameter estimate θ_2.

Confidence interval for θ_3:

The value of estimated coefficient θ_3 is $\hat{\theta}_3 = 0.013336833$ and standard error of the coefficient estimates of θ_3 is $SE(\hat{\theta}_3) = 0.001209831$.

Then a 95% confidence interval for θ_3 is as follows,

$$\hat{\theta}_3 - t_{0.025,51} \cdot SE(\hat{\theta}_1) \leq \theta_3 \leq \hat{\theta}_3 + t_{0.025,51} \cdot SE(\hat{\theta}_3)$$

$$0.013336833 - 2.007584 \times 0.001209831 \leq \theta_3 \leq 0.013336833 + 2.007584 \times 0.001209831$$

$$0.01090799 \leq \theta_3 \leq 0.01576567$$

Interpretation of confidence interval for θ_3:

Therefore, the interval from 0.01090799 to 0.01576567 forms a 95% confidence interval for the estimated coefficient θ_3. In other words, the interval from 0.01090799 to 0.01576567 gives the most believable value for the parameter estimate θ_3.

Table 6.9: Comparing the confidence interval for the estimated coefficients θ_1, θ_2, and θ_3 between the GNIM (unweighted nonlinear fit) and the WGNIM (weighted nonlinear fit).

parameters	GNIM		WGNIM	
	Lower Limit	**Upper Limit**	**Lower Limit**	**Upper Limit**
θ_1	0.08967951	0.24347389	0.0950086	0.177066574
θ_2	0.00382785	0.00650281	0.00345540	0.00593901
θ_3	0.00907756	0.01522246	0.01090799	0.01576567

Table 6.9 represents a comparison between the Weighted Gauss-Newton Iterative Method (WGNIM) and the Gauss-Newton Iterative Method (GNIM) with respect to the confidence intervals of the parameter estimates θ_1, θ_2, and θ_3. The findings suggest that, on average, the WGNIM produces more precise confidence intervals.

Table 6.10: Comparing the margin of error for the estimated coefficients θ_1, θ_2, and θ_3 between the GNIM (unweighted nonlinear fit) and the WGNIM (weighted nonlinear fit).

Parameters	GNIM	WGNIM
θ_1	**ME:** 0.07689718544	**ME:** 0.04102947804
θ_2	**ME:** 0.00133748478	**ME:** 0.00124180514
θ_3	**ME:** 0.00307245273	**ME:** 0.00242883736

Table 6.10 shows that in WGNIM the estimated coefficients θ_1, θ_2, and θ_3 have smaller margin of error in comparison to the GNIM, which is clearly shown in the Figure 6.25 and 6.26.

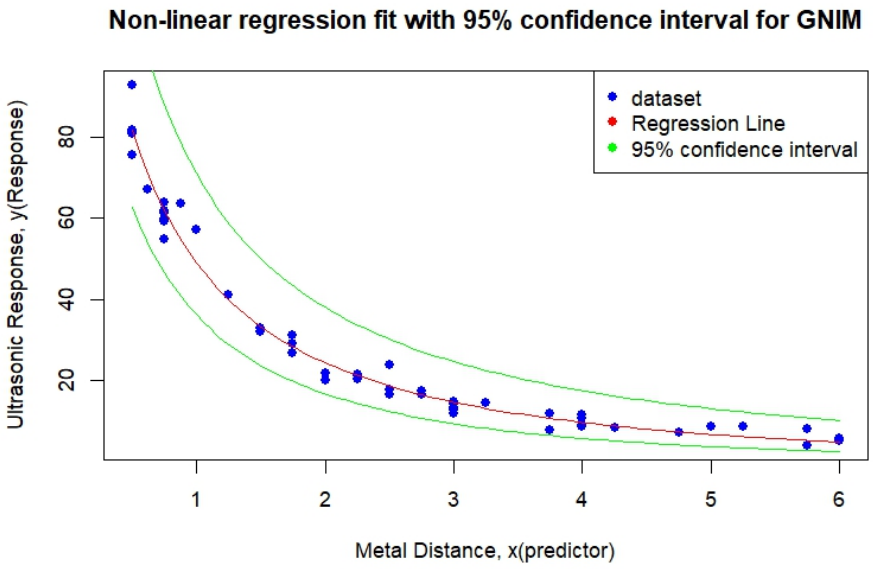


Figure 6.25: Graphical representation of nonlinear regression fit with 95% confidence interval for GNIM

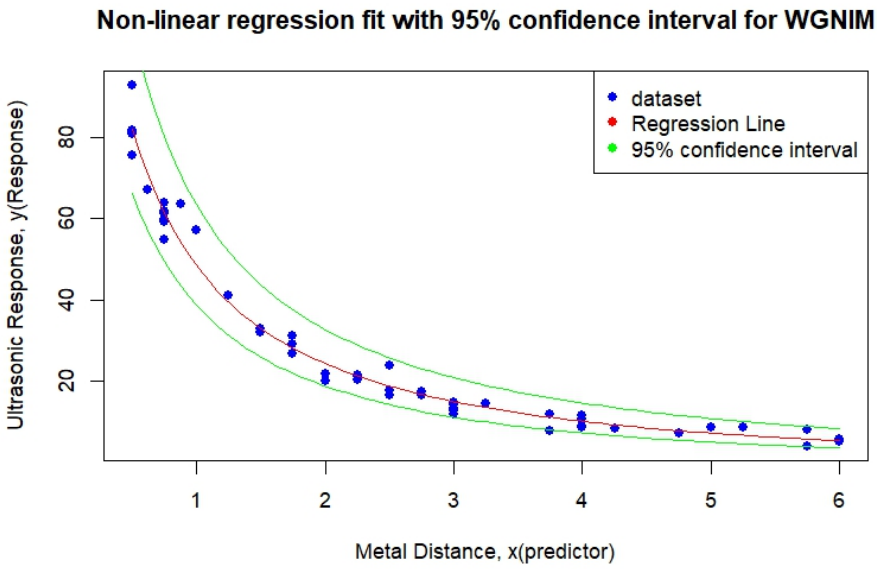


Figure 6.26: Graphical representation of nonlinear regression fit with 95% confidence interval for WGNIM

Smaller margin of error in WGNIM mean that the confidence interval for the estimated coefficient are narrower, which making the result of WGNIM more robust and trustworthy compared to GNIM. The WGNIM offers more accurate estimated coefficients, because it can take into consideration heteroscedasticity and assign appropriate weights to the observations during the fitting process.

6.2.5 Conclusions

In this study, classical statistical supervised learning optimization techniques like the Gauss-Newton Iterative Method (GNIM), the Weighted Gauss-Newton Iterative Method (WGNIM), the Reweighted Gauss-Newton Iterative Method (RGNIM), and the Levenberg-Marquart (LM) algorithm have been investigated and compared for resilience in multicollinarity and heteroscedasticity to fit nonlinear models. These methods are advanced extensions of the nonlinear least squares method that minimize the sum of squared differences

between the observed and predicted values of the nonlinear model.

The WGNIM method controls for heteroscedasticity by adjusting weights in the linearized model. The occurrence of structural multicollinearity, where parameter estimators have inflated variances, leads to inaccurate results for fitting the model. Multicollinearity is assessed via the Variance Inflation Factor (VIF). Under restricted levels of multicollinearity, the GNIM and RGNIM are examined and analyzed in simulation experiments. The results show that, with the occurrence of multicollinearity, the RGNIM does not outperform the regular GNIM for the logistic growth model and its corresponding particular dataset.

In application of the methods in this study, the association between metal distance (a predictor variable) and ultrasonic response (a response variable) is estimated in a NIST dataset from [15]. According to the results, heteroscedasticity is present because the variances were not constant. By including the estimation of weights for the given dataset in the GNIM, the WGNIM was able to solve this problem. The WGNIM effectively reduced the heteroscedasticity issue, resulting in constant variances, as demonstrated by the findings.

This improvement was noteworthy, as heteroscedasticity in the GNIM inflated regression coefficients, specifically an increase in the standard errors of the estimated coefficients, which in turn produced inflated confidence intervals. Nevertheless, the utilization of the WGNIM resolved this issue, and the confidence intervals were significantly better in comparison to those obtained with the GNIM. Compared to the GNIM, the WGNIM had a longer time required for each iteration as well as a longer total execution time. However, the WGNIM demonstrated efficiency by requiring fewer iterations to estimate the coefficients in comparison to the GNIM.

In a nutshell, better estimated coefficients and confidence intervals were the end result of using the WGNIM, which was shown to be an effective technique for handling heteroscedasticity in nonlinear regression models. Regardless, choosing between the GNIM and the WGNIM for similar applications requires careful consideration of the trade-off

between execution time and efficiency.

REFERENCES

[1] Lai, W. H., S. L. Kek, and K. G. Tay. 2017. *Solving nonlinear least squares problem using Gauss-Newton method.* International Journal of Innovative Science, Engineering & Technology. 4(1): 258-262.

[2] Sulaimon Mutiu, O. 2015. *Application of weighted least squares regression in forecasting.* International Journal of Recent Research in Interdisciplinary Sciences (IJRRIS). 2(3): 45-54.

[3] Zama, F. 2019. *Iteratively Reweighted Least Squares Algorithm for Nonlinear Distributed Parameter Estimation*. Annals of Reviews and Research. 5(2): 41-44.

[4] Shrestha, N. 2020. *Detecting multicollinearity in regression analysis.* American Journal of Applied Mathematics and Statistics. 8(2): 39-42.

[5] Gunst, R. F. and J. T. Webster. 1975. *Regression analysis and problems of multicollinearity.* Communications in Statistics-Theory and Methods. 4(3): 277-292.

[6] Marquardt, D.W. 1963. *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the Society for Industrial and Applied Mathematics. 11(2):431-441.

[7] Cook, R. D., and S. Weisberg. 1983. *Diagnostics for heteroscedasticity in regression.* Biometrika. 70(1): 1-10.

[8] Dette, H., & A. Munk. 1998. *Testing heteroscedasticity in nonparametric regression.* Journal of the Royal Statistical Society: Series B (Statistical Methodology). 60(4): 693-708.

[9] Klein, A. G., C. Gerhard, R. D. Büchner, S. Diestel, & K. Schermelleh-Engel. 2016. *The detection of heteroscedasticity in regression models for psychological data.* Psychological Test and Assessment Modeling. 58(4): 567.

[10] Levenberg, K. 1944. *A method for the solution of certain non-linear problems in least squares.* Quarterly of applied mathematics. 2(2): 164-168.

[11] Guillaume, P. and R. Pintelon. 1996. *A Gauss-Newton-like optimization algorithm for "weighted" nonlinear least-squares problems.* IEEE Transactions on Signal Processing, 44(9):2222-2228.

[12] Montgomery, D. C., E. A. Peck, and G. G. Vining. 2021. *Introduction to linear regression analysis.*John Wiley & Sons.

[13] Gareth, J., W. Daniela, H. Trevor, and T. Robert. 2013. *An introduction to statistical learning: with applications in R.* Spinger.

[14] Kutner, M. H., C. J. Nachtsheim, J. Neter, and W. Li. 2005. *Applied linear statistical models.* McGraw-hill.

[15] Chwirut, D. 1979. *chwirut2-Ultrasonic Reference Block Study.* NIST (National Institute of Standards and Technology(NIST).
`https://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`

[16] Nash, J. C. 2018. *Compact numerical methods for computers: linear algebra and function minimisation.* Routledge.

[17] Geman, S., Bienenstock, E., & Doursat, R. 1992. *Neural networks and the bias/variance dilemma*. Neural computation. 4(1): 1-58.

[18] Abraham, B., & Ledolter, J. 2004. *Introduction to Regression Modelling*. Thomson.

[19] Chatterjee, S., & Simonoff, J. S. 2013. *Handbook of regression analysis*. John Wiley & Sons.

[20] Shalab. 2017. *Regression Analysis, Chapter 3-Multiple Linear Regression Model*. IIT, Kanpur.

[21] Shalab. 2017. *Regression Analysis, Chapter 7-Generalized and Weighted Least Squares Estimations*. IIT, Kanpur.

[22] Kim, J. H. 2019. *Multicollinearity and misleading statistical results*. Korean journal of anesthesiology. 72(6): 558.

[23] Smyth, G. K. 2002. *Nonlinear regression*. Encyclopedia of environmetrics. 3: 1405-1411.

[24] Huang, H.H., Hsiao, C.K., Huang, S.Y. 2010. *International Encyclopedia of Education, Chapter: Nonlinear Regression Analysis*. Oxford: Elsevier. 339-346.

[25] Eitzen, D.G., Sushinsky, G.F., Chwirut, D.J. 1975. *Improved Ultrasonic Standard Reference Blocks*. In:NBS.

[26] Wang, S., Xu, M., Zhang, X., & Wang, Y. 2022. *Fitting Nonlinear Equations with the Levenberg–Marquardt Method on Google Earth Engine*. Remote Sensing. 14(9): 2055.

Appendix A

R CODE

A.1 GAUSS-NEWTON ITERATIVE METHOD

```
# Set options to print in decimal form and control the number of digits
options(scipen = 999, digits = 10)

x=c(0.5,1,1.75,3.75,5.75,0.875,2.25,3.25,5.25,0.75,1.75,2.75,4.75,0.625,
1.25,2.25,4.25,0.5,3,0.75,3,1.5,6,3,6,1.5,3,0.5,2,4,0.75,2,5,0.75,2.25,
3.75,5.75, 3, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.5, 6, 3, 0.5,
2.75,0.5,1.75)
ydata=c(92.9,57.1,31.05,11.5875,8.025,63.6,21.4,14.25,8.475,63.8,26.8,
16.4625,7.125,67.3, 41, 21.15, 8.175, 81.5, 13.12, 59.9,14.62,32.9,5.44,
12.56,5.44,32,13.95,75.8,20,10.42,59.5,21.67,8.55,62,20.2,7.76,3.75,11.81,
54.7,23.7, 11.55, 61.3, 17.7, 8.74, 59.2, 16.3, 8.62, 81, 4.87, 14.62,
81.7,17.17,81.3,28.9)

#-------------------------Gauss-Newton Method----------------------
maxi=1000
i=0
diff=1
beta_old=c(theta1=0.1, theta2=0.01, theta3=0.02)
# Define the nonlinear model
f=function(theta1, theta2, theta3) ((exp(-theta1*x)/(theta2+theta3*x)))
# Define the Jacobin matrix
jac=function(theta1, theta2, theta3)
                cbind((-x*exp(-theta1*x))/(theta2+theta3*x),
                  (-exp(-theta1*x))/(theta2+theta3*x)^2,
             (-x*exp(-theta1*x))/(theta2+theta3*x)^2)
# Store the solutions in the table
solutions_table <- matrix(0, nrow = maxi, ncol = 6)
```

```
colnames(solutions_table) <- c("Iteration", "Theta1", "Theta2", "Theta3",
"Convergence","Residual")

# Store the Jacobian matrices in a list
jacobian_list <- vector("list", length = maxi)

# ----------------Gauss-Newton method algorithm-----------------------

while (i<maxi && diff>0.000001) {
  j=jac(beta_old[1], beta_old[2], beta_old[3])
  res=c(ydata-f(beta_old[1], beta_old[2], beta_old[3]))
  residual=sum(res^2)

  #-------- Calculate the change in beta1 and beta2-------------
  solutions_table[i + 1, ] <- c(i + 1, beta_old[1], beta_old[2],
  beta_old[3], diff,residual)

  # -------Store the Jacobian matrix for this iteration---------
  jacobian_list[[i+1]] <- j
  #jacobian_list[[i + 1]] <- as.data.frame(j)
  colnames(jacobian_list[[i + 1]]) <- c("Z_i1", "Z_i2", "Z_i3")
  #Calculate the change in beta1 and beta2
  beta_new=beta_old+solve(t(j) %*% (j)) %*% t(j) %*% res
  #Increment the iteration counter
  i=i+1
  #Check if the solutions have converged
  diff=abs(sum((beta_old-beta_new)/beta_old))
  #Update the old parameter estimates
  beta_old=beta_new
}
#-----------------------------------------------------------------
```

```r
# Trim the solutions table to remove unused rows
solutions_table <- solutions_table[1:i, ]
#-----------------------------------------------------------------
#Print the solutions
cat("Theta1:", beta_old[1], "\n")
cat("Theta2:", beta_old[2], "\n")
cat("Theta3:", beta_old[3], "\n")
cat("Number of Iterations:", i, "\n")
cat("Final Difference in x:", diff, "\n")
#-----------------------------------------------------------------
#print the solution table
print(solutions_table)
#-----------------------------------------------------------------
# Print Jacobian matrices for each iteration
for (iter in 1:i) {
  cat("Iteration", iter, ":\n")
  print(jacobian_list[[iter]])
  cat("\n")
  cor(jacobian_list[[iter]])
  print(cor(jacobian_list[[iter]]))
  cat("\n")
}
# Reset options to default values
options(scipen = 0, digits = 6)
#-------------------------------------------------------------------
#Plot the regression line
plot(x, ydata, xlab = "Metal Distance, x(predictor)",
     ylab = "Ultrasonic Response, y(Response)",
     main="Non-linear fitting by Gauss Newton Method", pch=1, col="black")
curve((exp(-beta_old[1]*x)/(beta_old[2]+beta_old[3]*x)),
                          add = TRUE, col = "blue")
```

```
fitted_value <- exp(-beta_old[1]*x)/(beta_old[2]+beta_old[3]*x)
plot(fitted_value, res, xlab = "Fitted Value", ylab = "Residuals",
     main = "Plot of residuals vs fitted value for Gauss Newton Method")
abline(0,0)
```

A.2 WEIGHTED GAUSS-NEWTON ITERATIVE METHOD

```
# Set options to print in decimal form and control the number of digits
options(scipen = 999, digits = 10)

x=c(0.5,1,1.75,3.75,5.75,0.875,2.25,3.25,5.25,0.75,1.75,2.75,4.75,0.625,
1.25,2.25,4.25,0.5,3,0.75,3,1.5,6,3,6,1.5,3,0.5,2,4,0.75,2,5,0.75,2.25,
3.75,5.75, 3, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.5, 6, 3, 0.5,
2.75,0.5,1.75)
ydata=c(92.9,57.1,31.05,11.5875,8.025,63.6,21.4,14.25,8.475,63.8,26.8,
16.4625,7.125,67.3, 41, 21.15, 8.175, 81.5, 13.12, 59.9,14.62,32.9,5.44,
12.56,5.44,32,13.95,75.8,20,10.42,59.5,21.67,8.55,62,20.2,7.76,3.75,11.81,
54.7,23.7, 11.55, 61.3, 17.7, 8.74, 59.2, 16.3, 8.62, 81, 4.87, 14.62,
81.7,17.17,81.3,28.9)

##------------- Weighted Gauss-Newton Iterative Method--------------
maxi=1000
i=0
diff=1
beta_old=c(theta1=0.1, theta2=0.01, theta3=0.02)
# Define the nonlinear model
f=function(theta1, theta2, theta3) ((exp(-theta1*x)/(theta2+theta3*x)))
# Define the Jacobin matrix
jac=function(theta1, theta2, theta3)
        cbind((-x*exp(-theta1*x))/(theta2+theta3*x),
        (-exp(-theta1*x))/(theta2+theta3*x)^2,
        (-x*exp(-theta1*x))/(theta2+theta3*x)^2)
```

```
# Store the solutions in the table
solutions_table <- matrix(0, nrow = maxi, ncol = 6)
colnames(solutions_table) <- c("Iteration", "Theta1", "Theta2", "Theta3",
"Convergence","Residual")

# Store the Jacobian matrices in a list
jacobian_list <- vector("list", length = maxi)

df=data.frame(ydata,x)
dfs <- df[duplicated(df$x) | duplicated(df$x, fromLast = TRUE), ]
##  Determine Weights
dfs_1 = dfs[order(dfs$x),]
d = by(dfs_1$x,dfs_1$x,mean)
s2 = by(dfs_1$ydata,dfs_1$x,var)
md = as.vector(d)
vresp = as.vector(s2)
#lnmd = log(md)
#lnvresp = log(vresp)
#out2 = lm(lnvresp~lnmd)
#summary(out2)
out3 <- lm(vresp~md)
plot(md, vresp)
lines(md, fitted(out3))
summary(out3)
pr_v <- predict(out3,newdata=data.frame(md=x))
weight <- 1/pr_v

# -------------Weighted Gauss-Newton Iterative method----------------

while (i<maxi && diff>0.000001) {
```

```
  j=jac(beta_old[1], beta_old[2], beta_old[3])
  res=c(ydata-f(beta_old[1], beta_old[2], beta_old[3]))
  residual=sum(res^2)
  wt <- diag(weight, nrow = length(x), ncol = length(x))

  #-------- Calculate the change in beta1 and beta2-------------
  solutions_table[i + 1, ] <- c(i + 1, beta_old[1], beta_old[2],
  beta_old[3], diff,residual)

  # -------Store the Jacobian matrix for this iteration---------
  jacobian_list[[i+1]] <- j
  #jacobian_list[[i + 1]] <- as.data.frame(j)
  colnames(jacobian_list[[i + 1]]) <- c("Z_i1", "Z_i2", "Z_i3")
 #Calculate the change in beta1 and beta2
  beta_new=beta_old+solve(t(j)%*% wt %*% (j)) %*% t(j)%*% wt %*% res
 #Increment the iteration counter
  i=i+1
  #Check if the solutions have converged
  diff=abs(sum((beta_old-beta_new)/beta_old))
  #Update the old parameter estimates
  beta_old=beta_new
}
#-----------------------------------------------------------------
# Trim the solutions table to remove unused rows
solutions_table <- solutions_table[1:i, ]
#-----------------------------------------------------------------
#Print the solutions
cat("Theta1:", beta_old[1], "\n")
cat("Theta2:", beta_old[2], "\n")
cat("Theta3:", beta_old[3], "\n")
cat("Number of Iterations:", i, "\n")
```

```
cat("Final Difference in x:", diff, "\n")
#-----------------------------------------------------------------
#print the solution table
print(solutions_table)
#-----------------------------------------------------------------
# Print Jacobian matrices for each iteration
for (iter in 1:i) {
  cat("Iteration", iter, ":\n")
  print(jacobian_list[[iter]])
  cat("\n")
  cor(jacobian_list[[iter]])
  print(cor(jacobian_list[[iter]]))
  cat("\n")
}
# Reset options to default values
options(scipen = 0, digits = 6)
#-----------------------------------------------------------------
#Plot the regression line
plot(x, ydata, xlab = "Metal Distance, x(predictor)",
    ylab = "Ultrasonic Response, y(Response)",
    main="Non-linear fitting by Weighted Gauss Newton Method",
    pch=1, col="black")
curve((exp(-beta_old[1]*x)/(beta_old[2]+beta_old[3]*x)),
add = TRUE, col = "blue")
fitted_value <- exp(-beta_old[1]*x)/(beta_old[2]+beta_old[3]*x)
plot(sqrt(wt)*fitted_value, sqrt(wt)*res,
xlab = "Weighted Fitted Values",
ylab = "Weighted Residuals",
main="Plot of Residuals vs fitted values for Weighted Gauss Newton Method")
abline(0,0)
```

```
plot(md,vresp)
plot(md, fitted(out3))
```

A.3 NLS PACKAGE

```
# imports library
library(minpack.lm)
#library(nlsr)
#--------------------Gauss-Newton Iterative Method----------------------
x<-c(0.5,1,1.75,3.75,5.75,0.875,2.25,3.25,5.25,0.75,1.75,2.75,4.75,0.625,
1.25,2.25,4.25,0.5,3,0.75,3,1.5,6,3,6,1.5,3,0.5,2,4,0.75,2,5,0.75,2.25,
3.75,5.75, 3, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.5, 6, 3, 0.5,
2.75,0.5,1.75)
y <- c(92.9,57.1,31.05,11.5875,8.025,63.6,21.4,14.25,8.475,63.8,26.8,
16.4625,7.125,67.3, 41, 21.15, 8.175, 81.5, 13.12, 59.9,14.62,32.9,5.44,
12.56,5.44,32,13.95,75.8,20,10.42,59.5,21.67,8.55,62,20.2,7.76,3.75,11.81,
54.7,23.7, 11.55, 61.3, 17.7, 8.74, 59.2, 16.3, 8.62, 81, 4.87, 14.62,
81.7,17.17,81.3,28.9)

df=data.frame(y,x)
start_values <- c(theta_1=0.1, theta_2=0.01, theta_3=0.02)

m<-nlsLM(formula=y~(exp(-theta_1*x)/(theta_2+theta_3*x)),
         data=df,
         start=start_values,
         algorithm = "LM",
         control = nls.control(maxiter = 1000))
print(m)
coef(m)
summary(m)
fitted(m)
res = resid(m)
```

```
plot(x, ydata, xlab = "Metal Distance, Predictor (x)",
ylab = "Ultrasonic Response, Response (y)",
main = "Scatter plot of ultrasonic reference block data")
#lines(x, fitted(m), col="green")

plot(fitted(m), resid(m))
abline(0,0)

# Plot the data
plot(x, y, xlab = "Metal Distance, x (Predictor)",
ylab = "Ultrasonic Response, y (Response)")

# Generate x values for the curve
x_curve <- seq(min(x), max(x), length.out = 100)

# Predict y values using the fitted model
y_curve <- predict(m, newdata = data.frame(x = x_curve))

# Add the curve to the plot
lines(x_curve, y_curve, col = "red", lwd = 1)

#----------------Weighted Gauss-Newton Iterative Method-----------------

library(minpack.lm)
#library(nlsr)
x_1<-c(0.5,1,1.75,3.75,5.75,0.875,2.25,3.25,5.25,0.75,1.75,2.75,4.75,
0.625,1.25,2.25,4.25,0.5,3,0.75,3,1.5,6,3,6,1.5,3,0.5,2,4,0.75,2,5,0.75,
2.25,3.75,5.75, 3, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.75, 2.5, 4, 0.5, 6, 3,
0.5,2.75,0.5,1.75)
y_1 <- c(92.9,57.1,31.05,11.5875,8.025,63.6,21.4,14.25,8.475,63.8,26.8,
    16.4625,7.125,67.3, 41, 21.15, 8.175, 81.5, 13.12, 59.9,14.62,32.9,
```

```
    5.44,12.56,5.44,32,13.95,75.8,20,10.42,59.5,21.67,8.55,62,20.2,7.76,
    3.75,11.81,54.7,23.7, 11.55, 61.3, 17.7, 8.74, 59.2, 16.3, 8.62, 81,
    4.87, 14.62, 81.7,17.17,81.3,28.9)

df=data.frame(y_1,x_1)
start_values <- c(theta1=0.1, theta2=0.01, theta3=0.02)

dfs <- df[duplicated(df$x_1) | duplicated(df$x_1, fromLast = TRUE), ]
##  Determine Weights
dfs_1 = dfs[order(dfs$x_1),]
d = by(dfs_1$x_1,dfs_1$x_1,mean)
s2 = by(dfs_1$y_1,dfs_1$x_1,var)
md = as.vector(d)
vresp = as.vector(s2)
#lnmd = log(md)
#lnvresp = log(vresp)
#out2 = lm(lnvresp~lnmd)
#summary(out2)
out3 <- lm(vresp~md)
summary(out3)
pr_v <- predict(out3,newdata=data.frame(md=x_1))
weight <- 1/pr_v

m_1<-nlsLM(formula=y_1~(exp(-theta1*x_1)/(theta2+theta3*x_1)),
          data=df,
          start=start_values,
          weights = weight,
          algorithm = "LM",
          control = nls.control(maxiter = 1000))

print(m_1)
```

```
coef(m_1)
summary(m_1)
fitted(m_1)
resid(m_1)
#plot(x, ydata)

# Generate x values for the curve
x_curve <- seq(min(x_1), max(x_1), length.out = 100)

# Predict y values using the fitted model
y_curve <- predict(m_1, newdata = data.frame(x_1 = x_curve))

# Add the curve to the plot
lines(x_curve, y_curve, col = "green", lwd = 1)
      legend("topright", legend = c("Gauss Newton Iterative Method",
      "Weighted Gauss Newton Iterative Method"),
       col = c("red", "green"), lwd = 1)
title("Non-linear Fitting with Gauss Newton Iterative Method ", line = 2)
title("and Non-linear Fitting using Weighted Gauss Newton Iterative Method ",
line = 1)
plot(fitted(m_1), resid(m_1))
abline(0,0)
```

A.4 GAUSS-NEWTON ITERATIVE METHOD (WITHOUT WEIGHT) FOR LOGISTIC GROWTH MODEL

```
#Load_Data-----------------------------------------------------------
x=c(1,2,3,4,5,6,7,8,9,10,11,12)
y=c(5.308, 7.24, 9.638, 12.866, 17.069, 23.192, 31.443,
    38.558, 50.156, 62.948, 75.995, 91.972)
#Initial starting value----------------------------------------
```

```
beta_old=c(theta1=200, theta2=50.50, theta3=0.3035)
theta1=beta_old[1];
theta2=beta_old[2];
theta3=beta_old[3];
# Define the nonlinear model----------------------------------------
f=function(theta1, theta2, theta3) {(theta1/(1+theta2*exp(-theta3*x)))
}
# Define the Jacobin matrix-----------------------------------------
jac=function(theta1, theta2, theta3){
cbind("Z_1"=1/(1+theta2*exp(-theta3*x)),
"Z_2"= (-theta1*exp(-theta3*x))/(1+theta2*exp(-theta3*x))^2,
"Z_3"=(theta1*theta2*x)/(exp(theta3*x)*(1+theta2*exp(-theta3*x))^2))
}
jac(theta1, theta2, theta3)
#Residual function-------------------------------------------------
Res=function(theta1, theta2, theta3){ y-f(theta1, theta2, theta3)
}
Res(theta1, theta2, theta3)
#Jacobian inverse and matrix poduct---------------------------------
#Find the Jacobians and matrix product with weights
winvjac= function(theta1, theta2, theta3, w){
  J=jac(theta1, theta2, theta3);
  R=Res(theta1, theta2, theta3);
  solve(t(J)%*%w%*%J )%*% t(J)%*%w%*%R
}
#Find  the weights at each iteration-------------------------------
Warray=array(rep(0), dim = c(n, n, nsim))
#w=matrix(rep(0), nrow = n, ncol = p)
#function for weights of a linear regression model
resw=function(H){
  rg=lm(Res(theta1, theta2, theta3)~H[,2]+H[,3]+H[,4], data=H);
```

```
  resandfitted=lm(resid(rg)~rg$fitted, data = H )
  gw=1/(resid(resandfitted) )^{2};
  gw
}
nsim=5
beta_new=matrix(rep(0), nrow = length(beta_old), ncol =nsim )
for (j in 1:nsim) {
  beta_old=c(theta1=200, theta2=50.50, theta3=0.3035);
  theta1=beta_old[1];
  theta2=beta_old[2];
  theta3=beta_old[3];
  H=cbind(Res(theta1, theta2, theta3), data.frame(IJac[,,j]));
  w=diag(resw(H));
  Warray[,,j]=w;
  beta_new[,j]=beta_old+winvjac(theta1, theta2, theta3, w);
  theta1=beta_new[1,j];
  theta2=beta_new[2,j];
  theta3=beta_new[3,j];
  beta_old=beta_new[,j]
}
#produce weights at each iteration
Warray
#The estimates for Theta at each iteration.
beta_new
B2=beta_new;
# Jacobian at each iteration
#sample size
n=length(y)
#number of parameters p = number of Z_1, Z_2, Z_3..
p=3
#IJac==matrix of jacobian iterations.
```

```
IJac=array(rep(0), dim = c(n,p,nsim ))
for (j in 1:nsim) {
  theta1=beta_new[1,j];
  theta2=beta_new[2,j];
  theta3=beta_new[3,j];
  IJac[,,j]=jac(theta1, theta2, theta3)
}
IJac

#Variance inflation factor at each iteration---------------
library(car)
library(carData)
library(usdm)
VIFMatrix=matrix(rep(0), nrow = p, ncol = nsim)
for (j in 1:nsim) {
  r=data.frame( IJac[,,j]);
  a=data.frame( vif( r ));
  VIFMatrix[,j]=a$VIF
}
VIFMatrix
#-------------------------------------------------------------
```

A.5 REWEIGHTED GAUSS-NEWTON ITERATIVE METHOD FOR LOGISTIC GROWTH MODEL

```
# Re-weighted least squares
#Datasets
#-----------------------------------------------------
x=c(1,2,3,4,5,6,7,8,9,10,11,12)
y=c(5.308, 7.24, 9.638, 12.866, 17.069, 23.192, 31.443,
    38.558, 50.156, 62.948, 75.995, 91.972)
#Starting Values
```

```
#-----------------------------------------------------
beta_old=c(theta1=200, theta2=50.50, theta3=0.3035)
theta1=beta_old[1];
theta2=beta_old[2];
theta3=beta_old[3];
# Define the nonlinear model
#-----------------------------------------------------
f=function(theta1, theta2, theta3) {
            (theta1/(1+theta2*exp(-theta3*x)))
}
# Define the Jacobin matrix
#-----------------------------------------------------
jac=function(theta1, theta2, theta3){
  cbind("Z_1"=1/(1+theta2*exp(-theta3*x)),
"Z_2"= (-theta1*exp(-theta3*x))/(1+theta2*exp(-theta3*x))^2,
"Z_3"=(theta1*theta2*x)/(exp(theta3*x)
                            *(1+theta2*exp(-theta3*x))^2))
}
jac(theta1, theta2, theta3)
#Residual function
#-----------------------------------------------------
Res=function(theta1, theta2, theta3){
           y-f(theta1, theta2, theta3)
}
Res(theta1, theta2, theta3)
#Jacobian inverse and matrix poduct
#Find the Jacobians and matrix product with weights
winvjac= function(theta1, theta2, theta3, w){
  J=jac(theta1, theta2, theta3);
  R=Res(theta1, theta2, theta3);
  solve(t(J)%*%w%*%J )%*% t(J)%*%w%*%R
```

```
}
#Find  the weights at each iteration

Warray=array(rep(0), dim = c(n, n, nsim))
#w=matrix(rep(0), nrow = n, ncol = p)
#function for weights of a linear regression model
resw=function(H){
  rg=lm(Res(theta1, theta2, theta3)~H[,2]+H[,3]+H[,4], data=H);
  resandfitted=lm(resid(rg)~rg$fitted, data = H )
  gw=1/(resid(resandfitted) )^{2};
  gw
}
nsim=5
beta_new=matrix(rep(0), nrow = length(beta_old), ncol =nsim )
for (j in 1:nsim) {
  beta_old=c(theta1=200, theta2=50.50, theta3=0.3035);
  theta1=beta_old[1];
  theta2=beta_old[2];
  theta3=beta_old[3];
  H=cbind(Res(theta1, theta2, theta3), data.frame(IJac[,,j]));
  w=diag(resw(H));
  Warray[,,j]=w;
  beta_new[,j]=beta_old+winvjac(theta1, theta2, theta3, w);
  theta1=beta_new[1,j];
  theta2=beta_new[2,j];
  theta3=beta_new[3,j];
  beta_old=beta_new[,j]
}
###produce weights at each iteration
Warray
#The estimates for Theta at each iteration.
```

```
beta_new
B2=beta_new;
# Jacobian at each iteration
#sample size
n=length(y)
#number of parameters p = number of Z_1, Z_2, Z_3..
p=3
#IJac==matrix of jacobian iterations.
IJac=array(rep(0), dim = c(n,p,nsim ))
for (j in 1:nsim) {
  theta1=beta_new[1,j];
  theta2=beta_new[2,j];
  theta3=beta_new[3,j];
  IJac[,,j]=jac(theta1, theta2, theta3)
}
IJac
```