COMPARISON OF ITERATIVE ALGORITHMS FOR PARAMETER ESTIMATION IN NONLINEAR REGRESSION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GAMZE MUSLUOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
STATISTICS

SEPTEMBER 2018

Approval of the thesis:

COMPARISON OF ITERATIVE ALGORITHMS FOR PARAMETER ESTIMATION IN NONLINEAR REGRESSION

submitted by **GAMZE MUSLUOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Statistics Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Ayşen Dener Akkaya
Head of Department, **Statistics** _____

Prof. Dr. Ayşen Dener Akkaya
Supervisor, **Statistics Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil
Statistics Dept., METU _____

Prof. Dr. Ayşen Dener Akkaya
Statistics Dept., METU _____

Assoc. Prof. Dr. Özlem Türker Bayrak
Inter Curricular Courses Dept., Çankaya University _____

Date: 07.09.2018

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Gamze Musluoğlu

Signature :

ABSTRACT

COMPARISON OF ITERATIVE ALGORITHMS FOR PARAMETER ESTIMATION IN NONLINEAR REGRESSION

Musluoğlu, Gamze
MSc., Department of Statistics
Supervisor : Prof. Dr. Ayşen Dener Akkaya

September 2018, 104 pages

Nonlinear regression models are more common as compared to linear ones for real life cases e.g. climatology, biology, earthquake engineering, economics etc. However, nonlinear regression models are much more complex to fit and to interpret. Classical parameter estimation methods such as least squares and maximum likelihood can also be adopted to fit the model in nonlinear regression as well, but explicit solutions can not be achieved unlike linear models. At this point, iterative algorithms are utilized to solve the problem numerically. Since there is no extensive study which compiles, classifies and compares the existing methods for nonlinear parameter estimation, the objective of this study is to fill this gap. In our study, we aim to compile the methods which are used for nonlinear parameter estimation purpose and compare them with respect to several criteria such as bias, execution time, number of iterations etc. The comparison will be conducted considering different scenarios which are small vs. large sample sizes, good vs. poor initial values, normal vs. non-normal error terms, simple vs complex models (with respect to number of parameters), and robustness. Both real and simulated data are used in the comparative study.

Keywords: Nonlinear Regression, Iterative Algorithms, Nonlinear Least Squares, Nonlinear Parameter Estimation

ÖZ

DOĞRUSAL OLMAYAN REGRESYONUN PARAMETRE TAHMİNİNDE KULLANILAN TEKRARLI ALGORİTMALARIN KARŞILAŞTIRILMASI

Musluoğlu, Gamze
Yüksek Lisans, İstatistik
Tez Yöneticisi : Prof. Dr. Ayşen Dener Akkaya

Eylül 2018, 104 sayfa

Doğrusal olmayan regresyon modelleri, gerçek hayat problemlerinde doğrusal modellere oranla daha yaygındır. İklim bilimi, biyoloji, deprem mühendisliği ve ekonomi örnekler arasındadır. Ancak doğrusal olmayan regresyon modellerini kurmak ve yorumlamak daha zordur. En küçük kareler ve en çok olabilirlik yöntemleri gibi klasik parametre tahmin yöntemleri, doğrusal olmayan regresyon için de kullanılmaktadır. Ancak doğrusal modellerde olduğu gibi kesin sonuçlara ulaşılamamaktadır. Bu noktada bu sorunu numerik olarak çözmek için tekrarlı algoritmalar kullanılır. Bu yöntemleri derleyen, sınıflandıran ve karşılaştıran geniş çaplı bir çalışma olmadığı için, amacımız bu boşluğu doldurmaktır. Bu çalışmada, doğrusal olmayan regresyon modellerinin tahmininde kullanılan yöntemleri derleyip, onları yanlılık, yürütme zamanı, tekrar sayısı gibi kriterlere göre karşılaştırmayı amaçlamaktayız. Karşılaştırma küçük ve büyük örneklem büyüklüğü, iyi ve kötü parametre başlangıç değerleri, normal ve normal olmayan hata terimleri, basit ve karmaşık modeller (parametre sayısına göre) ve sağlamlık gibi farklı senaryolar üzerinden uygulanacaktır. Karşılaştırmalı çalışmada hem gerçek hem de benzetim yolu ile elde edilmiş veri kullanılacaktır.

Anahtar Kelimeler: Doğrusal Olmayan Regresyon, Tekrarlı Algoritmalar, Doğrusal Olmayan En Küçük Kareler, Doğrusal Olmayan Parametre Tahmini

To My Family

ACKNOWLEDGMENTS

Firstly, I would like to express my gratitude to my supervisor Prof. Dr. Ayşen Dener Akkaya for her everlasting support on this work. She has been more than a supervisor to me. I could not be able to complete this thesis without her guidance.

I also would like to thank my examining committee members Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil, Assoc. Prof. Dr. Özlem Türker Bayrak and Prof. Dr. Ashis SenGupta for their valuable time and feedback.

My sincere gratitude is for my bestfriends Begüm Yentür, Seren Ergan, Nilay Kılıç Ateş, Deniz Çelikel, Kaan Uyanık and Orçun Denemeç who helped me to get through this process and listened to me whenever I am in need. Additionally, I am also grateful for the guidance and support provided my dear co-workers Ezgi Ayyıldız, Buket Coşkun, Duygu Varol and Elif Akça.

Finally, I want to thank my family, Hakan Musluoğlu, Sema Musluoğlu, Burcu Musluoğlu and my fiancé Ali Berkcan Boylu. I cannot find the suitable words to express how lucky I feel for having them by my side during this study. Thank you all.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

BFGS	Broyden-Fletcher-Goldfrab-Shanno
CG	Conjugate Gradient
DFP	Davidon-Fletcher-Powell
FR	Fletcher-Reeves
GL	Generalized Logistic
LS	Least Squares
L-M	Levenberg-Marquardt
ML	Maximum Likelihood
MSE	Mean Squared Error
NIST	National Institute of Standards and Technology
NLS	Nonlinear Least Squares
StRD	Statistical Reference Datasets

CHAPTER 1

INTRODUCTION

Nature in general is nonlinear. That is the main reason of the popularity of nonlinear regression in many different research fields. Nonlinear regression analysis is a commonly used tool for explaining the relationship between a set of variables by constructing a plausible model which contains nonlinear terms. It has applications in countless research areas such as physics, biology, earthquake engineering, economics etc. However, it has several difficulties in application such as model specification, assignment of starting values for parameters and more importantly estimation of the unknown model parameters which is the main focus for this study.

To estimate the model parameters and approximate a model fit, nonlinear least squares procedure is the most commonly used method, but it is not straightforward as it is in linear case. At this point, optimization algorithms offer help to overcome the inconvenience of the equations resulting from the nonlinear least squares procedure. Each method has different properties with pros and cons and there are numerous algorithms for this purpose, in the literature. Hence, the user face with a serious challenge when in the need of nonlinear regression analysis for his/her specific case. At this point, comparative studies are very useful to help the user to be able to choose the most suitable method for his/her case.

The fundamental to many numerical algorithms is Newton's method of Isaac Newton and it was first published in 1771 officially even though composed in 1685. His version included the calculation of a sequence of complex polynomials to obtain an approximation of the root of interest. Raphson (1697) reviewed the Newton's method and used successive approximations of the root rather than polynomials.

Throughout the following years, many researchers based their proposed algorithms on this essential method. Gauss (1809) introduced his updated recursion formula which drops the second derivative part of Newton-Raphson algorithm. In 1909, Debye introduced the method of steepest descent which uses the negative gradient as search direction and a reasonable step size to facilitate fast convergence. Both Gauss-Newton and steepest descent method had strenghts and weaknesses so, Levenberg (1944) merged their strong aspects to form a new and better algorithm. Based on Levenberg's study, Marquardt proposed an improved formula for the procedure in 1963. In 1964, Fletcher and Reeves took the idea of the famous conjugate gradient method to generalize it for nonlinear equations as well. Their method is called as nonlinear conjugate gradient method in the literature. Besides all these gradient-based algorithms, John Nelder and Roger Mead (1965) suggested their derivative-free algorithm. Following these advances, quasi-newton algorithms were suggested, which is a modification to fundamental Newton-Raphson method. This family of methods uses different formulas to obtain an approximation to the inverse Hessian matrix. Most popular ones were proposed by Broyden et al. (1970) and Davidon et al. (1991).

As mentioned briefly, numerous numerical algorithms exist for the purpose of parameter estimation in nonlinear regression analysis. Yet, in the literature, there is almost no comparative study which covers the most commonly used numerical algorithms for the solution of nonlinear least squares procedure. The present studies in the literature only focus on the performance of algorithms or the software packages under ideal conditions. They do not take the possible scenarios into account such as non-normality in the distribution of the error terms, selection of initial values, sample size, complexity of the model function and robustness.

The aim of this study is to compare the existing and commonly used numerical algorithms for nonlinear least squares problems under several conditions which can be summarized as the conditions with the presence of normal or non-normal errors, small or large sample size, selection of initial values for the iterations, simple or complex models and robustness. As comparison criteria, bias and mean squared error for the

estimates of the model parameters, mean squared error of fit, execution time and number of iterations will be used.

In this thesis, we make an introduction to the nonlinear regression analysis and explain the difficulties related to it in Chapter 2. One of these difficulties is parameter estimation problem which is our primary interest in this study. In Chapter 3, the methods used for parameter estimation in nonlinear regression are introduced. These methods are the most commonly used ones obtained through an extensive literature review. The results of the comparisons for iterative methods based on both real data and simulation study are given in Chapter 4. Finally, we summarize and conclude our findings in Chapter 5.

CHAPTER 2

OVERVIEW OF NONLINEAR REGRESSION ANALYSIS

Regression analysis is a useful and commonly known tool which is used for describing and modeling the relationship between variables. More specifically, it aims to examine the dependency of an exploratory (or response) variable and explanatory (or predictor) variables. It enables one to understand how response variable changes when the one of the predictor variables is varied whereas the other predictor variables are fixed. To sum up, the regression analysis is useful for

i. Examining the effect of the predictors on the response variable.
ii. Testing whether the predictors do well in estimating the response variable.

To construct a regression model thoroughly, one has to find the suitable regression function which fits well to the data. Regression function can be linear or nonlinear depending on the problem and the data.

When the model is linear, the procedure is called as linear regression analysis and classical model function is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \in_i , \quad i = 1,2, \ldots . n \tag{2.1}$$

where

Y_i: response variable

$X_{ij}'s$: regressor variables

$\beta_j's$: regression parameters ; j=1,2,…p

$\in_i$: random error.

Linear regression analysis is a commonly used regression analysis which is very useful in many statistical problems. It's preferred due to its simplicity, yet efficiency and statistical power. However, sometimes linear regression may not be appropriate due to intrinsic nonlinear pattern in relations between parameters or variables and the response. At this point, nonlinear regression analysis introduces the necessary complexity. In such cases, it is more realistic to use nonlinear regression analysis since many real life data do not follow a straight line pattern. Nonlinear data can be encountered in many real life problems in various research fields like attenuation relationships in earthquake engineering, tsunami modeling, weather forecasting in climatology, growth of a plant in agriculture etc.

Nonlinear regression model is generally given by

$$Y_i = f(X_i; \beta) + \in_i \qquad (2.2)$$

where

Y_i: response or dependent variable

$f(X_i; \beta)$: model function whose at least one of the derivatives with respect to unknown parameters contains at least one regression parameter, i.e., nonlinear in variables or parameters.

$\in_i$: random error.

Nonlinearity can be considered in terms of variables and parameters. Nonlinearity in variables can be handled by various transformation techniques. Here is a very simple example for this type of nonlinear models.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \in_i \tag{2.3}$$

In the model equation above, nonlinearity arises from X^2 and one can simply define a new variable, say $Z = X^2$. Carrying out these type of classical transformations, the models can be converted into a linear one. After that, the same procedure can be followed as in linear regression. There is nothing complex with these models. However, the second type of models are nonlinear in parameters and this type of nonlinearity can cause some complexity in solving and commenting on the problem.

Models that are nonlinear in parameters can also be considered in two different forms. The first one is called as intrinsically linear models which means they can be transformed to linear models.

As an example, Cobb-Douglas production model can be shown.

$$Y = \alpha\, L^{\beta} K^{\gamma} * \in_i \tag{2.4}$$

where

Y: response variable (output)

L, K: regressor variables (labor and capital, respectively)

α, β, γ: regression parameters

$\in_i$: random error.

The model in Equation 2.4 is clearly nonlinear in parameters, but it can be linearized by using log-transformation. After log-transformation, the Equation 2.5 is obtained.

$$\ln(Y) = \delta + \beta \ln(L) + \gamma \ln(K) \tag{2.5}$$

In the transformed model equation, δ stands for $\ln(\alpha)$ and other notations are the same. The natural logarithm versions of the variables can be substituted with Y^*, L^* and K^*, respectively. As it can be clearly seen, the resulting model is linear in its parameters and ready to be solved by applying classical linear regression analysis procedure.

On the other hand, the second type that we will mention about is intrinsically nonlinear models. They can not be transformed into a linear model unlike the first type. One example for them is as below.

$$Y = \frac{\beta_1}{\beta_2} exp\left[\frac{-(X-\beta_3)}{2\beta_2^2}\right] + \in_i \tag{2.6}$$

The model in Equation 2.6 is intrinsically nonlinear and should be treated as so. Thus, the researcher should adopt nonlinear regression methods to estimate its parameters. In this study, the main focus will be on estimation of intrinsically nonlinear models.

Though nonlinear regression models are very useful tools in solving complex relations, it has some difficulties in practice. The important and problematic issues to be careful about when conducting nonlinear regression analysis will be explained briefly in the following sections.

2.1 Model Specification

Specification of the model in nonlinear regression analysis is a crucial step. Model specification is about specifying the expectation function for the model and features of the error term. Specification of the error term is also important because assumptions about the error usually gives a direction to the researcher to conduct the analysis properly. By direction, parameter estimation technique is referred.

In general, nonlinear models arise as solutions to differential equations. On the other hand, the expectation function does not have to be in an explicit function form. For example, compartmental type of models have an expected response which is a solution to a set of differential equations.

There are variety of nonlinear models and their application areas can differ. It is not very easy to choose the best model from a large list of nonlinear regression functions. Before choosing the model, field of the research should be considered. In other words, similar studies in the literature can be examined to be able to choose the right family of models for the case. Moreover, plotting the data can be useful to see the structure of the model.

Since it is not our main scope, we will just give some examples to most common nonlinear regression models. Growth models are probably the most commonly used type of nonlinear models. Growth models are usually used to illustrate how response variable grows due to the changes in explanatory variable(s) (Montgomery et al., 2012). Its fields of applications are extensive from engineering to the sciences. Growth models are utilized to model the growth of an organism or bacteria in biology, growth of animals, humans or even growth of economy etc. As mentioned, its applications are extensive. Some of the commonly used growth models are explained briefly as follows:

Logistic Growth Model

$$Y_i = \frac{\alpha}{1+exp(-\beta-\delta t_i)} + \in_i \quad (2.7)$$

In the model (2.7), Y is the response variable and t is explanatory variable. α , β and δ are the unknown parameters for the given model. The parameters have different explanations for different cases. In order to accommodate the parameters well, the data should be understood clearly.

Exponential Growth Model

In the model (2.8), Y is the response variable while t is the explanatory variable. Y_0 is the starting value of the response when t=0 and k is the growth rate parameter. Interpretation of the parameters changes from case to case.

$$Y_i = Y_0 \exp(kt_i) + \in_i \quad (2.8)$$

Gompertz Growth Model

Many re-parameterizations can be found in the literature for Gompertz growth function. One and commonly used one is illustrated as in (2.9) (Tjorve and Tjorve,2017).

$$Y_i = \alpha . exp\big(-exp(-k(t_i - \beta))\big) + \in_i \quad (2.9)$$

2.2 Parameter Estimation

In contrast to simplicity of linear regression, things get a little complex for the nonlinear regression models. Parameter estimation is not very easy since the model is nonlinear in parameters and classical estimation techniques cannot provide exact solutions. Similar to linear regression, parameter estimation can still be done by following the least squares (LS) or maximum likelihood (ML) estimation procedures though there are some differences. The objective function for the nonlinear model is also nonlinear and to solve it, partial derivatives which are not easy to deal with in most of the cases are necessary. Even if the partial derivatives are here, the findings are not linear and the explicit solutions cannot be obtained directly. Moreover, making comments on the parameter estimates becomes confusing. Thus, use of iterative methods become inevitable in parameter estimation.

2.3 Initial Values

As mentioned earlier, to solve the parameter estimation problems in nonlinear regression analysis, numerical algorithms are adopted to reach the optimal value. Numerical algorithms are iterative which means the estimate is updated until the convergence to the best possible solution is achieved. Hence, it should have some starting value(s) to initiate the iterative process.

The goodness of initial values is highly important for the success of the numerical method because poor assignment of initial values can result in very slow convergence, convergence to local optima or no convergence at all. On the other hand, good starting point gives the right direction to the algorithm and facilitate its performance in many aspects. That is why assigning starting values well is an essential step in parameter estimation problems in nonlinear regression.

There are many possible options to find good starting values. The first one is plotting the data and interpreting the structure. It is simple yet very efficient way to decide on the initial values. Another one is guessing it based on the interpretation of the parameter of interest. For some models, model parameters have very clear meanings and one can simply decide on the initial guess for them with the help of his/her knowledge related to the field of subject. For instance, Fekedulegn, Siurtain and Colbert (1999) discussed the nonlinear models used in agricultural research and stated that the parameters in these models are meaningful in the field of agriculture, which forms a basis to obtain good initial values. Even some formulas which provide good initial guesses for some parameters are presented in the study. However, this only applies to those specific family of models. In most of the cases, there are no such formulas for initial value assignment. Third way to determine the initial values is using linearization or stochastic algorithms such as genetic algorithm. They perform quite well in this.

Since initial guesses play an important role on the performance of an algorithm in converging to right optimal value, we want to test our selected algorithms both under poorly and well selected initial values to see their robustness to it.

CHAPTER 3

ITERATIVE METHODS FOR THE PARAMETER ESTIMATION IN NONLINEAR REGRESSION

In this chapter, focus is on the parameter estimation methods commonly used in nonlinear regression analysis. Mainly, there are two estimation methods that is used for this purpose, least squares and maximum likelihood. However, maximum likelihood eventually becomes a least squares problem with both normal and non-normal errors (Seber and Wild, 2003). That is why we only consider least squares approach in this study.

3.1 Nonlinear Least Squares Estimation

Least squares principle is widely used for analyzing both linear and nonlinear models. The basic idea behind the LS principle is minimizing the sum of squares of deviations between the observed response variable Y and the fitted model value $\hat{Y}$.

Let $\in$ is used to denote the error term, the objective function S is obtained as follows:

$$S(\beta) = \sum_{i=1}^{n}(Y_i - f(X_i;\hat{\beta}))^2 = \sum_{i=1}^{n} r_i^2 \qquad (3.1)$$

where

Y: response variable

$f(X,\beta)$: regression model function

r_i: residual.

Having obtained the objective function S(β), the procedure is carried out by taking the derivatives of the objective function S(β) with respect to unknown parameters ($\beta_i's$) one by one. For each unknown parameter, a normal equation $g(\beta_i) = \frac{\partial S(\beta)}{\partial \beta_i} = 0$ is obtained. Then, the corresponding estimate results are found by solving these normal equations.

Application of LS principle to nonlinear models provides nonlinear set of normal equations which are difficult to solve with simple algebra operations. Because majority of nonlinear models does not have analytical solutions, iterative techniques become necessary. In the following section, most commonly used iterative methods developed and used to obtain nonlinear least sqaures (NLS) parameter estimates will be presented.

3.2 Numerical Methods Used in Nonlinear Least Squares Estimation

Numerical algorithms are countless and they have similar working principles more or less. Most of them starts with an initial guess and updates the current estimate depending on its specific search direction and step size. The iteration process continues until the convergence criterion is satisfied. Most of the time, convergence criterion is the difference between the successive iterative values of the estimated parameters or the corresponding value of the objective function. When one of them is smaller than ε which is a pre-defined very small quantity (e.g. 10^{-5} for this study.), the iterative process comes to an end. It can be illustrated as below where k denotes the iteration number.

$$S(\beta^{k+1}) - S(\beta^k) < \varepsilon \tag{3.2}$$

or

$$\beta^{k+1} - \beta^k < \varepsilon \tag{3.3}$$

In other words, iteration continues improving the fit until there is no significant change. Some algorithms may provide slow convergence while another one converges in few iterations. Each algorithm has different properties and different attitudes towards the problem.

3.2.1 Newton-Raphson Method

Newton-Raphson method, i.e. Newton's method, is probably the most well-known numerical algorithm used for nonlinear parameter estimation. The method forms a basis to many other optimization algorithms in the literature. It is named after Isaac Newton (1685) and Joseph Raphson (1697). The method tries to find an optimal solution iteratively. To be able to do this, it only requires a starting value for the algorithm and partial derivatives of the objective function. The basic idea behind Newton-Raphson method is starting with a reasonable initial guess and successively obtaining a better approximation to the root of the function of interest.

Newton-Raphson method is based on the quadratic approximation to the objective function. The quadratic approximation is as follows where ∇ denotes the derivative operator.

$$S(\beta^k + \delta^k) = S(\beta^k) + \nabla S(\beta^k)^T \delta^k + \frac{1}{2} {\delta^k}^T H_k \delta^k \tag{3.4}$$

where

β^k: current estimate

$\delta^k, {\delta^k}^T$: step size at the kth iteration and its transpose

$\nabla S(\beta^k)$, $\nabla S(\beta^k)^T$: first derivative of the model function at the current estimate and its transponse

H_k: second derivative at the current estimate (Hessian matrix).

The recursion formula obtained from this approximation is

$$\beta^{k+1} = \beta^k - \frac{\nabla S(\beta^k)}{\nabla^2 S(\beta^k)} \tag{3.5}$$

or equivalently,

$$\beta^{k+1} = \beta^k - H^{-1}(\beta^k)\, g(\beta^k). \tag{3.6}$$

In the recursion formula, some notations are used and they can be expressed as

$$\nabla S(\beta^k) = g(\beta^k) = \sum_{i=1}^{n} R_i(\beta^k)\, \nabla R_i(\beta^k) = J(\beta^k)^T R(\beta^k) \tag{3.7}$$

$$\nabla^2 S(\beta^k) = H(\beta^k) = \sum_{i=1}^{n} \nabla R_i(\beta^k)\, \nabla R_i(\beta^k)^T + \sum_{i=1}^{n} R_i(\beta^k) \nabla^2 R_i(\beta^k) \tag{3.8}$$

$$= J(\beta^k)^T J(\beta^k) + A(\beta^k) \tag{3.9}$$

where

$g(\beta^k)$: gradient (first derivative) of the function at β^k

$H(\beta^k)$: Hessian evaluated at β^k

$J(\beta^k)$: jacobian evaluated at β^k

$R(\beta^k)$: residuals evaluated at β^k

$A(\beta^k)$: second derivative part of the Hessian matrix.

Another way to show the recursion formula of Newton-Raphson is as following.

$$\beta^{k+1} = \beta^k + [J(\beta^k)^T J(\beta^k) + A(\beta^k)]^{-1} J(\beta^k)^T R(\beta^k) \qquad (3.10)$$

Newton-Raphson method is a powerful method with a quadratic convergence characteristic. Quadratic convergence is that as the convergence to the root occurs, the difference between the root itself and the approximation is squared at each step. So, it doubles the number of significant digits in every step. However, Newton-Raphson method does not converge if certain conditions do not hold. The assumptions of quadratic convergence proof should be met to guarantee that convergence will occur for any specific function. The assumptions can be listed as follows:

i. $f'(\beta) \neq 0$; for all β .
ii. $f''(\beta)$ is continuous.
iii. β_0 is reasonably close to the true optimal value β^*.

In the assumptions above, $f'(\beta)$ and $f''(\beta)$ denote the first and second derivatives of the model function, respectively. Additionally, β_0 refers to a set of initial values for the model parameters.

In numerical analysis, it is common that the algorithm may not converge even in large number of iterations. As it is highlighted in the third assumption, the convergence of the algorithm is highly dependent to the choice of initial values for model parameters. In other words, Newton-Raphson method has a very small region of convergence. It works very well with good initial values which are very close to the solution.

On the other hand, bad initial guesses can lead to non-convergence of the algorithm. As a result, it is very crucial to choose good initial values with small error. As stated earlier, Newton-Raphson method is a fundamental method which is a basis for many numerical algorithms suggested and studied in the literature. Some of the methods we use in our study are modifications to this method and their difference will be explained in detail.

3.2.2 Gauss-Newton Method

Gauss-Newton is a very commonly used optimization method for solving nonlinear least squares problems. The method is also known as "linearization method". Gauss-Newton method is a modification of the classical Newton's method. It simply ignores the second derivative part $A(\beta)$ and this simplifies the recursion formula to solve. The idea behind that elimination is that the second derivatives are assumed to be highly small compared to first derivatives. That is why they differ only slightly with respect to the derivative matrix that is used in computations. In many cases, they converge to very close estimates.

The convergence rate can change from case to case. It may converge slowly or not converge at all. The reason behind the convergence problem should be investigated and solved to obtain realistic solutions. In general, linear convergence is expected against Newton-Raphson's quadratic convergence feature. The performance of Gauss-Newton method is highly related to whether the second derivative part is important or not. Here, importance refers to magnitude of the quantity. If it is not that important, even quadratic convergence can be achieved.

It is a simple procedure compared to the other iterative schemes because it makes use of only first derivative. Yet, it is efficient and convergent. Moreover, not using second derivatives saves both time and storage.

The procedure simply uses the first-order Taylor series expansion. The method utilizes a linear approximation to the regression function around the preset initial values of the unknown parameters and it iteratively improves the estimate.

$$S(\beta^k + \delta^k) = S(\beta^k) + \nabla S(\beta^k)^T \delta^k \tag{3.11}$$

And Equation 3.11 is equivalent to

$$S(\beta^{k+1}) = S(\beta^k) + \nabla S(\beta^k)\,(\beta^{k+1} - \beta^k) \tag{3.12}$$

which leads to

$$\beta^{k+1} - \beta^k = \delta^k = [J(\beta^k)^T J(\beta^k)]^{-1} J(\beta^k)^T R(\beta^k). \tag{3.13}$$

As a result, the following recursion formula is obtained.

$$\beta^{k+1} = \beta^k + [J(\beta^k)^T J(\beta^k)]^{-1} J(\beta^k)^T R(\beta^k) \tag{3.14}$$

It continues until the convergence criterion mentioned in Equation 3.2 is satisfied. Convergence can take long in some cases, but still for many real life cases it is very useful.

3.2.3 Steepest Descent Method

Steepest descent method, or gradient descent method, is an optimization tool that is used for minimization of the function of interest. The basic procedure starts with deciding on the initial points and then taking the gradient of the model function at the preset initial value(s). After obtaining the gradient of the function, the solution is moved in the negative direction of the gradient and each time this happens, convergence criterion is checked to see whether the optimal solution is obtained or not. The process is repeated until convergence criterion is satisfied. The algorithm will eventually converge when the gradient is 0 or very close to 0. The convergence criterion needs to be selected beforehand. It is a very small value but larger than 0. To summarize we can show the algorithm step by step.

Let S be the objective function that we want to minimize, which is convex and differentiable. Also let us denote the unknown parameter vector as θ.

i. Choose set of initial values β^k (k=0, initially) and decide on the convergence criterion.
ii. Take the gradient of the function that we want to minimize and evaluate it in the set of initial values that is selected before. So, let $g_k = \nabla S(\beta^k)$ and give start to the iteration process.
iii. Set the search direction as the negative of the gradient evaluated at the current point β^k.
iv. Choose a step size λ to use in the iteration process as a fixed value or choose a different step size for each iteration which is called as adaptive step size.
v. The updated estimate is obtained with $\beta^{k+1} = \beta^k - \lambda \nabla S(\beta^k)$. Check if the convergence is achieved. If not, repeat the process starting from the $2^{nd}\ step$ by setting k=k+1.
vi. Repeat the process until convergence happens.

One aspect to take into consideration while using this algorithm is estimating the step size. It is very crucial because it may lead to divergence or slow convergence. Slow convergence is not a desirable feature and when step size is too small it will take a long time to convergence. When step size is too large iterations may diverge. Selection of step size can be carried out with different methods in the literature such as line search (Curry, 1944).

Steepest descent has a branch called steepest ascent. As understood by its name, it is used for maximizing the objective function. In other words, while steepest descent looks for the global minimum, steepest ascent searches the global maximum for a particular problem. The algorithm follows the same steps as steepest descent with only one difference. Unlike steepest descent, the solution is stepped in the positive direction of the gradient of the function. This is because it aims to converge to the global maximum. The recursion formula can be illustrated as in the Equation 3.15.

$$\beta^{k+1} = \beta^{k} + \lambda \nabla f(\beta^{k}) \tag{3.15}$$

In the formula above, β^{k+1} is the updated estimate while β^{k} is the current estimate. As mentioned, only difference from the steepest descent is the direction that will be followed in each iteration. All other main characteristics of steepest descent method apply to steepest ascent, too.

3.2.4 Levenberg-Marquardt Method

Levenberg-Marquardt (L-M) method is one of the most famous methods used in nonlinear parameter estimation. The method is developed by Marquardt as a modification to the Gauss-Newton method in 1963. Since his work is based on a work conducted by Levenberg (1944), the method is also called as Marquardt's compromise method in the literature.

In some problems, the conditions for $J^T J$ may not hold. In other words, the matrix may be singular or ill-conditioned. In such situations, Gauss-Newton algorithm may not converge at all. Therefore, L-M method is developed as a modification to the Gauss-Newton method to overcome the non-convergence problem of Gauss-Newton when the Jacobian matrix is singular in some cases.

L-M method combines the features of Gauss-Newton method and the steepest descent method altogether. The method of steepest descent tends to work well in early iterations but as it comes closer to the optimal solution, starts to work slowly. On the contrary, the situation is the complete opposite for Gauss-Newton method. So, as a combination of these features, L-M method uses the strategy of steepest descent in early iterations and then switches to Gauss-Newton as it gets closer to the end.

It makes use of the linearization vector of Gauss-Newton and the direction of gradient descent method (Montgomery et al., 2012).

Based on that, Levenberg (1944) suggested a modification to the Gauss-Newton algorithm with the increment

$$\beta^{k+1} - \beta^k = \delta^k_{Levenberg} = [J(\beta^k)^T J(\beta^k) + \eta\, I]^{-1} J(\beta^k)^T R(\beta^k) \qquad (3.16)$$

where I denotes the identity matrix.

Moreover, η is called as conditioning factor and it plays a very important role in convergence. As η goes to infinity the step becomes the steepest descent step and as η goes to 0 it looks like Gauss-Newton step. This factor is used for adjustment.

Based on his idea, Marquardt (1963) suggested a modification to Levenberg's increment which is given by

$$\beta^{k+1} - \beta^k = \delta^k_{Marquardt} = [J(\beta^k)^T J(\beta^k) + \eta\, D]^{-1} J(\beta^k)^T R(\beta^k) \qquad (3.17)$$

where D is a diagonal matrix that usually consists of the diagonal entries of the matrix $J^T J$.

About the rate of convergence, L-M is known to have quadratic convergence feature when the jacobian at the current point is nonsingular. This is a very good characteristic because quadratic convergence refers to a very fast convergence. However, there are many problems that have problematic jacobians. In that case, its convergence property deteriorates to linear convergence.

So, it can be concluded that implementation of L-M method is more complex than the Gauss-Newton method, in general. There are modifications in the literature for this method, too (see for example, Fan (2011)).

3.2.5 Quasi-Newton Methods

Quasi-Newton family of methods are generalization of classical Newton's method. Newton's method is known to be very efficient and fast when necessary conditions hold. But when one of them is absent, the method fails to converge properly. One of these assumptions is the presence of second derivatives of the objective function. However, second derivatives are not always easy to handle or even sometimes they are not obtainable at all. For such cases, quasi-Newton method approximates the inverse of the Hessian matrix at each iteration. This feature is one of the advantages of quasi-Newton methods over Newton's method. In Newton's method, after calculating Hessian matrix, its inverse should be obtained as a next step. Instead, quasi-Newton directly approximates the inverse Hessian matrix, which makes the whole procedure a more easier.

Approximation of the inverse Hessian matrix is conducted by analyzing the successive gradients obtained from the same function of interest. In short, this modification is carried out either to simplify the procedure of step direction calculation or not to calculate second derivatives.

As in Newton's method, quasi-Newton algorithms uses the quadratic approximation, too. Its quadratic approximation to the objective function is as following:

$$S(\beta^k + \delta^k) = S(\beta^k) + \nabla S(\beta^k)^T \delta^k + \frac{1}{2} {\delta^k}^T H_k \delta^k \qquad (3.18)$$

where

β^k: current estimate

δ^k: step size at the k^{th} iteration

$\nabla S(\beta^k)$: first derivative at the current estimate

H_k: second derivative at the current estimate (Hessian matrix).

General iterative scheme for quasi-Newton algorithms is

$$\beta^{k+1} = \beta^k - \alpha_k {H_k}^{-1} g_k \tag{3.19}$$

where

H_k: symmetric and nonsingular approximated Hessian matrix

g_k: gradient

α_k: step size obtained by line search.

The procedure is the same until the Hessian matrix calculation because quasi-Newton methods do not directly calculate it, rather approximate it. In the literature, many different approximation formulas for the inverse Hessian is suggested. The first quasi-Newton algorithm belongs to William C. Davidon (1959) and then the method he proposed was updated by Fletcher and Powell (1963). The method named after these three statisticians as DFP method. As years go by, many modifications were developed and became very popular. Commonly known modifications can be listed as Broyden-Fletcher-Goldfarb-Shanno (Broyden et al., 1970), Davidon–Fletcher–Powell (Davidon, 1991), Simple Rank 1 (Byrd, 1996) and Broyden's method (Broyden,1965).

In this study, BFGS and DFP updating formulas are included due to their popularity. They will be explained briefly in the following sections.

3.2.5.1 Davidon–Fletcher–Powell Method

The Davidon–Fletcher–Powell (DFP) method is the first member of quasi-Newton family. It was first suggested by Davidon (1959) and his idea was improved by another study (Fletcher and Powell, 1963). Even if many other algorithms were developed in

the literature through the years, DFP method is still one of the best quasi-Newton algorithms.

Again the basic principle is straightforward, only the Hessian approximation procedure is different. It offers updating formulas for both Hessian or inverse Hessian directly. The latter is mostly preferred in practice.

The approximation to Hessian matrix is conducted with the Equation 3.20.

$$H_{k+1} = \left(I - \frac{\varphi_k {\delta^k}^T}{{\varphi_k}^T \delta^k}\right) H_k \left(I - \frac{\delta^k {\varphi_k}^T}{{\varphi_k}^T \delta^k}\right) + \frac{\varphi_k {\varphi_k}^T}{{\varphi_k}^T \delta^k} \tag{3.20}$$

where

$$\varphi_k = \nabla S(\beta^k + \delta^k) - \nabla S(\beta^k)$$

and

$$\gamma_k = \frac{1}{{\varphi_k}^T \delta^k}.$$

As stated, the method also suggests a formula for direct approximation to the inverse Hessian matrix as in the Equation 3.21.

$$B_{k+1} = B_k + \frac{\delta^k {\delta^k}^T}{{\delta^k}^T \varphi_k} - \frac{B_k \varphi_k {\varphi_k}^T B_k}{\varphi_k B_k {\varphi_k}^T} \tag{3.21}$$

The approximation is iteratively updated at each iteration. Another well-working method from quasi-Newton family will be explained in the following section.

3.2.5.2 Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method

Broyden-Fletcher-Goldfarb-Shanno **(**BFGS) algorithm is one of the most popular quasi-Newton methods which is used for nonlinear unconstrained optimization problems. The method is developed by Broyden, Fletcher, Goldfarb and Shanno (1970).

The approximation formula for Hessian is shown as in the Equation 3.23.

$$H_{k+1} = H_k + \frac{\varphi_k {\varphi_k}^T}{{\varphi_k}^T \delta^k} - \frac{H_k \delta^k {\delta^k}^T {H_k}^T}{{\delta^k}^T H_k \delta^k} \tag{3.23}$$

The method also offers a direct approximation to inverse Hessian as in the Equation 3.24.

$$B_{k+1} = \left(I - \frac{\delta^k {\varphi_k}^T}{{\varphi_k}^T \delta^k}\right) B_k \left(I - \frac{\varphi_k {\delta^k}^T}{{\varphi_k}^T \delta^k}\right) + \frac{{\delta^k}^T \delta^k}{{\varphi_k}^T \delta^k} \tag{3.24}$$

where

$$\varphi_k = \nabla S(\beta^k + \delta^k) - \nabla S(\beta^k) \tag{3.25}$$

or equivalently

$$\varphi_k = \nabla S(\beta^{k+1}) - \nabla S(\beta^k) \tag{3.26}$$

which is basically the difference between successive gradients. The approximation for Hessian is updated at each iteration based on the previous approximation.

The above mentioned quasi-Newton methods are widely applicable for unconstrained minimization problems and proven to be efficient. Since nonlinear parameter estimation is a special case of unconstrained optimization, these methods are included in our study.

3.2.6 Nonlinear Conjugate Gradient Method

Conjugate gradient (CG) is a numerical algorithm which is primarily used for large system of linear equations. The method was first introduced by Eduard Stiefel and Magnus Hestenes (1952). Although its main aim is not minimization, it is being used for that purpose as well.

Conjugate gradient has a nonlinear version which is utilized to find the optimal value in problems with nonlinear equations and it is called as nonlinear conjugate gradient method. It makes use of the gradient of the objective function. The algorithm is explained through the following steps:

i. Start with obtaining the classical steepest descent direction which is simply the negative gradient of the function.

$$\delta^k = -\nabla S(\beta^k) \tag{3.27}$$

ii. Calculate the conjugate direction C_k by using one of the formulas suggested in the literature. Commonly used ones can be listed as Fletcher-Reeves, Polak–Ribière and Hestenes-Stiefel. Our choice is Fletcher-Reeves method.

$$C_k^{FR} = \frac{{\delta^k}^T \delta^k}{{\delta^{k-1}}^T \delta^{k-1}} \tag{3.28}$$

iii. Obtain the conjugate direction S_k using the findings and update it at each iteration.

$$S_k = \delta^k - C_k S_{k-1} \tag{3.29}$$

iv. Find a reasonable step size α_k by performing line search. Here, argmin function finds the arguments that minimize the objective function.

$$\alpha_k = argmin\, S(\beta^k + \alpha S_k) \tag{3.30}$$

v. Use the recursion formula to update the current estimate

$$\beta^{k+1} = \beta^k + \alpha_k S_k \tag{3.31}$$

vi. Continue until the convergence criterion is satisfied.

3.2.7 Nelder-Mead Method

Nelder-Mead algorithm is a well-known derivative-free method which is commonly used in unconstrained optimization problems. The method is also known as simplex search algorithm and it was suggested by John Nelder and Roger Mead (1965).

It does not require the calculation or approximation of the derivatives, so it is applicable to non-differentiable functions as well. Its working principle is based on simplex. The method makes use of simplex, which is a representation of triangle in random dimensions. For example, a simplex has n+1 vertices when it is defined for a problem in $\mathbb{R}^n$ and it is called as n-simplex. According to that, when we are working with 2 dimensional space, the simplex has 3 vertices. When the dimension increases to 3, number of vertices becomes 4. The shapes of the mentioned simplexes can be observed in Figure 3.1.

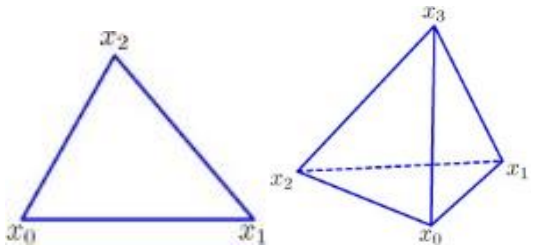

Figure 3.1: 2-simplex (triangle) and 3-simplex (tetrahedron)

Algorithm for the Nelder-Mead method can be summarized as follows.

i. Construct an initial working simplex S. There exist numerous ways to obtain an initial simplex and most commonly used technique is generating n+1 vertices around the pre-defined starting value.

ii. Check the convergence criterion. If not satisfied yet, transform the working simplex. Four transformation approaches are available, namely, shrinkage, expansion, reflection and contraction. The suitable one is applied to obtain the updated simplex.

iii. Repeat the process until convergence to optimal solution is achieved. Convergence criterion is up to the researcher however it is assumed that the convergence is achieved when there is a very slight difference between the successive simplexes.

Despite its simplicity in the idea and application, Nelder-Mead method is still considered as very successful in practice. As a result, it is popular and preferred by researchers from many different fields. Moreover, according to many sources, it is among the best known derivative-free algorithms. Hence, Nelder-Mead method takes part in our study on behalf of derivative-free methods.

CHAPTER 4

SIMULATION STUDY AND APPLICATION

4.1 Organization of the Simulation Study

In this section, simulation study is designed and carried out to compare the performances of the selected numerical algorithms under several conditions. The conditions taken into consideration are distribution of the error terms, sample size, goodness of initial guesses for parameters, complexity of the model and robustness.

For this purpose, two models are selected: a simple model and a complex one. The complexity of the model is based upon the number of parameters that the model consists of.

The simulation study is performed for each combination of the possible scenarios that we are interested in. Additionally, these simulations are executed for both simple and complex models.

Codes for the Monte Carlo simulation study is written in the R program by using built-in functions for the numerical algorithms. One of the codes is given in Appendix part as an example.

In the following sections, organization of the simulation study will be explained in detail and the results will be illustrated.

4.1.1 Conditions for Comparison

In this study, the aim is to compare the iterative methods in an extensive way which means not only under ideal conditions, but also under poor conditions. There are many conditions that may affect the performance of the methods. As statisticians, the first one that comes up to mind is the non-normality in the distribution of the error terms. In addition to non-normality, robustness of the estimated parameters to anomalies in the error distribution, i.e. outliers, contamination and inliers are checked. Another condition is directly related to optimization problems, i.e. initial values used to start the iteration process. Moreover, sample size is considered as a significant factor effecting the performance of the algorithms because it does on regression. Finally, the number of parameters which can affect the performance of the iterative algorithms is also examined. As a result, the simulation study is conducted for each one of the combinations of all these conditions, separately. That allows us to make comparison for any case, very clearly.

4.1.1.1 Complexity of the Model

Complexity of the model is one aspect that is taken into consideration through this study. Here the word "complexity" refers to the number of parameters included in the nonlinear regression model. In other words, higher the number of parameters is, more complex the model is.

In nonlinear regression models, number of parameters does not directly depend on the number of variables unlike linear models. Hence, the number of parameters are presumed as reference to describe the complexity of the model. When the model gets more complex, it is naturally harder to estimate the parameters. Since we want to test the most commonly used algorithms under both situations, one simple and one complex model are selected.

The models are taken from National Institute of Standards and Technology (NIST) Statistical Reference Datasets (StRD) from the website http://www.itl.nist.gov/div898/strd/. The website offers numerous models with their reference datasets which are classified according to their level of difficulty as low, medium and high. We have selected one model with low level of difficulty and another with high level of difficulty.

Another advantage of this website is that it also offers the certified values for the parameters. The certified values are reliable because they were approved by using at least two numerical methods and different software packages. Presence of certified values helps us to compare the outcomes that will be obtained from each algorithm we will make use of.

Simple Model

The simple model is named as "Chwirut1" and taken from an ultrasonic response study (Chwirut, 1979). In the model, the dependent variable Y is ultrasonic response, and the independent variable is metal distance. Its reference dataset has 214 observations and 3 unknown parameters. The data is observed, not simulated.

The model function is given by

$$Y_i = \frac{e^{-\beta_1 X_i}}{\beta_2 + \beta_3 X_i} + \epsilon_i \tag{4.1}$$

where β_1, β_2 and β_3 are unknown parameters and ϵ_i is the error term.

The certified values for the unknown parameters are as following:

$\beta_1^* = 0.1902$, $\beta_2^* = 0.0061$, $\beta_3^* = 0.0105$.

The behavior of the relationship can be observed from Figure 4.1. As it can be clearly seen, it has a decreasing trend with a curvature shape.

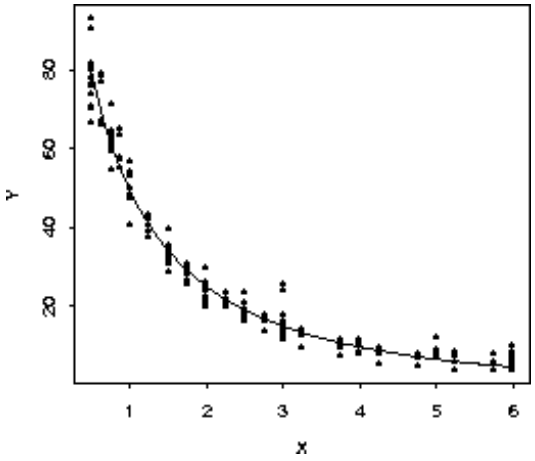

Figure 4.1: The nonlinear regression curve of Chwirut1

Complex Model

Second model is named as "Thurber" and taken from a semiconductor electron mobility study conducted by Thurber (1979). It is called as complex model in this study due to its higher number of parameters as compared to the former model. Moreover, it is also classified as a difficult model and dataset in the website according to different aspects as well.

The model has one response and one predictor variable again. They are the measure of electron mobility and the natural log of the density, respectively. The data consists of 37 observations.

The model function has 7 unknown parameters and is given by

$$Y_i = \frac{\beta_1+\beta_2 X_i+\beta_3 {X_i}^2+\beta_4 {X_i}^3}{1+\beta_5 X_i+\beta_6 {X_i}^2+\beta_7 {X_i}^3} + \epsilon_i \qquad (4.2)$$

where β_1, β_2, β_3, β_4, β_5, β_6 and β_7 are the unknown parameters included in the model of interest and ϵ_i is the error term.

In addition, the certified values for these parameters are as follows:

$\beta_1^* = 1288.139$, $\beta_2^* = 1491.079$, $\beta_3^* = 583.238$, $\beta_4^* = 75.416$, $\beta_5^* = 0.966$,

$\beta_6^* = 0.397$, $\beta_7^* = 0.049$

Moreover, its curvy behavior can be observed in Figure 4.2.

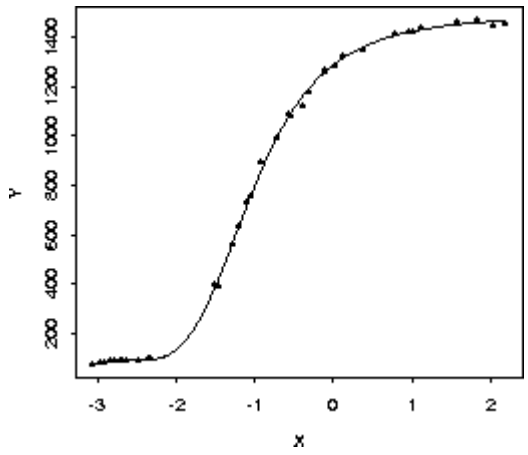

Figure 4.2: The nonlinear regression curve of Thurber

4.1.1.2 Distribution of the Error

Error term is anything that differentiates the constructed model from its true version and has always been the problematic part of any problem. Errors are often assumed as normally distributed since it is the easiest way to run away from the problems that may occur. On the other hand, normality assumption is hard to satisfy in real life problems. However, researchers insist on believing that their error terms are normally distributed. In fact, it can be any distribution and the assumption of normality may cause problems. That is why it is included in this comparative study. The aim here is to notice which one(s) of the numerical algorithms overcome this inconvenience more efficiently.

We have selected the generalized logistic (GL) distribution as a non-normal distribution for the error terms in this study. The probability distribution of generalized logistic is given by

$$f(e) = \frac{\frac{b}{\sigma} exp(^{-e}/_{\sigma})}{\{1+exp(^{-e}/_{\sigma})\}^{b+1}} \qquad -\infty < e < \infty \tag{4.3}$$

with the following properties

$$E(e) = \{\Psi(b) - \Psi(1)\}\,\sigma \tag{4.4}$$

$$V(e) = \{\Psi'(b) + \Psi'(1)\}\,\sigma^2 \tag{4.5}$$

where b>0, $\Psi(x)$ denotes the psi-function and $\Psi'(x)$ stands for its derivative.

In GL distribution, the shape parameter b is crucial. When it is equal to 1, the distribution is classical logistic distribution. On the other hand, it is negatively skewed when b<1 and positively skewed when b>1 (Akkaya and Tiku, 2010). Hence, it can behave in any direction. That is why we have selected it as our non-normal distribution and assign b=0.5 and b=2 to observe the performance results under both types of skewness.

4.1.1.3 Goodness of Initial Values

As mentioned in the Section 2.3, goodness of initial values is one of the most crucial aspects to take into consideration when the study involves optimization (Bates and Watts, 2007). Since nonlinear parameter estimation is a special case of unconstrained optimization, obtaining good starting values is one of the best steps that the researcher can take to conduct nonlinear regression analysis successfully.

Respectively good starting guesses facilitates the convergence by leading the iteration process in the right direction. Here, "good" refers to the value close to the real value of the parameter. In our case, real value refers to the certified values provided by the website itself. On the other hand, when the starting values are respectively poor, i.e. farther from the certified values, iteration process can head to wrong direction and end up with non-convergence, or it converges to optimal value very slowly. These cases are not desired, so selection of good starting values should be emphasized.

It is always easier to converge when the process is initiated with a set of good starting values. An efficient method should give plausible results under both conditions. That is why the behavior of algorithms are tested under the presence of poor initial values as well as good ones.

To select starting values efficiently, there are numerous ways as mentioned briefly in Section 2.3. In this study, website from which the reference models taken provides 3 different sets of starting values which are close to the certified ones. So, we have taken one of them as our set of good starting values. Based on that, a set of poor starting values are derived by increasing the values of good ones with the ratio 50%.

4.1.1.4 Sample Size

Sample size is another issue that is important in any kind of regression analysis. Regression analysis tries to explain the relationship between variables by constructing a meaningful model. At this point, as sample size gets larger, the information it carries gets bigger and this effects the model in a positive way. An estimator showing this behavior is called as consistent. As a result, by performing a nonlinear regression analysis, sample size issue is considered as a test condition.

During comparisons, n=250 and n=25 are taken as quite large and small sample sizes, respectively to notice their distinctive effect on success of our numerical algorithms.

4.1.1.5 Robustness in Errors

Deviations from the assumed error distribution is highly frequent in many cases. As a result, it is almost never for sure that the assumed distribution is fully accurate (Tiku & Akkaya, 2004). To overcome this successfully, robustness of the estimators comes into question. The estimators should be able to carry on its efficiency for an assumed distribution and maintains high efficiency even in such deviations from the distribution to be called as robust.

To test the robustness of the estimators of the variables and the model itself, we designed outlier, contamination and inlier models for the cases in which the errors are normally distributed.

Outlier Model

$(n - r)\ X_i$ come from $N(\mu, \sigma^2)$ and r (we do not know which) come from $N(\mu, 4\sigma^2)$; $r = [0.5 + 0.1n]$ (integer value) and μ=0 and σ=1 without any loss of generality. In other words, the standard deviation is doubled up for 10% of the observations (Tiku and Akkaya, 2004).

Inlier Model

In the inlier model, a proportion of smallest or largest order statistics in a random sample $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ from $N(\mu, \sigma^2)$ is replaced by $\bar{\varepsilon} + (-\delta)^i \sigma, \ |\delta| \leq 1,$ so that the displaced observations get located closer to $\bar{\varepsilon}$ and are erroneous, and located within $\bar{\varepsilon} \pm \sigma$ (Akkaya and Tiku, 2008).

Contamination Model

In this case, the normal error distribution is contaminated with another distribution which is uniform (0, 1) with 10 percent, i.e., 0.90N(μ, σ)+0.10Uniform(0, 1), and μ=0 and σ=1 without any loss of generality (Tiku and Akkaya, 2004).

4.1.2 Simulation Scenarios

The conditions discussed in Section 4.1 form the base for this comparative study. Since the aim is to test the iterative methods under many conditions, all possible combinations of these conditions are assigned as a simulation scenario which increases our cases to 24 for each model, i.e., simple and complex models. The point is that an undesirable condition does not always occur when all other conditions are well-behaved. All undesirable conditions can arise together or one at a time. That is why the simulation study is designed by crossing over the levels of selected conditions.

This allows us to observe the behavior of the method under a wide range of situations from best to the worst case possible.

Our simulation scenarios are presented as below:

1. Large sample + normally distributed errors + good initial values
2. Small sample + normally distributed errors + good initial values
3. Large sample + GL (b=0.5) distributed errors + good initial values
4. Small sample + GL (b=0.5) distributed errors + good initial values
5. Large sample + GL (b=2) distributed errors + good initial values
6. Small sample + GL (b=2) distributed errors + good initial values
7. Large sample + normally distributed errors + poor initial values
8. Small sample + normally distributed errors + poor initial values
9. Large sample + GL (b=0.5) distributed errors + poor initial values
10. Small sample + GL (b=0.5) distributed errors + poor initial values
11. Large sample + GL (b=2) distributed errors + poor initial values
12. Small sample + GL (b=2) distributed errors + poor initial values
13. Large sample + normally distributed errors with outliers + good initial values
14. Small sample + normally distributed errors with outliers + good initial values
15. Large sample + normally distributed errors with outliers + poor initial values
16. Small sample + normally distributed errors with outliers + poor initial values
17. Large sample + contaminated normal errors + good initial values
18. Small sample + contaminated normal errors + good initial values
19. Large sample + contaminated normal errors + poor initial values
20. Small sample + contaminated normal errors + poor initial values
21. Large sample + normally distributed errors with inliers + good initial values
22. Small sample + normally distributed errors with inliers + good initial values
23. Large sample + normally distributed errors with inliers + poor initial values
24. Small sample + normally distributed errors with inliers + poor initial values

These simulations scenarios are carried out both for simple model and complex models, separately. So, it is possible to compare the results of similar conditions with respect to the complexity of the model.

4.1.3 Comparison Criteria

To compare the success of commonly used numerical methods for estimating the unknown parameters of a nonlinear regression model and reaching a reliable conclusion, comparison should be based upon reliable criteria. The criteria used in this comparative study are explained briefly in the following subsections.

4.1.3.1 Bias

The first criterion is bias in the estimators and is widely applicable in estimation comparison problems. When evaluating the performance of a numerical algorithm, it is highly crucial to obtain the closest estimate to the true value. It is basically the difference between the estimated value and the true value of the parameter of interest.

$$Bias = E(\hat{\beta}) - \beta \tag{4.6}$$

In Equation 4.6, $\hat{\beta}$ is the parameter estimate and β is the true value of the parameter itself.

4.1.3.2 Mean Squared Error

Mean squared error (MSE) is the second criterion for our comparison study. It is a measure of assessment for an estimator. In our study, the purpose is to see which algorithm(s) provides the best estimates for the unknown parameters. That is why MSE is selected as a criterion for comparison.

MSE is defined as

$$MSE = E\left[\left(\hat{\beta} - \beta\right)^2\right]. \tag{4.7}$$

It can also be written as the summation of variance of the estimated value and squared bias of it.

$$MSE = Var\left(\hat{\beta}\right) + Bias\left(\hat{\beta}\right)^2 \tag{4.8}$$

For real data case, variances are computed with the following formula.

$$Var\left(\hat{\beta}\right) = S^2\,[J'J]^{-1} \tag{4.9}$$

where

$$S = \sqrt{\frac{\sum_{i=1}^{n}[Y_i - f(X_i,\beta)]^2}{n-p}}. \tag{4.10}$$

4.1.3.3 Execution Time

Execution time is another criterion for comparing the performance of numerical algorithms. It is a measurement of speed, clearly. In fact, almost all numerical methods are very fast and they differ from each other with milliseconds sometimes. However, in this work this criterion is still preferred for assessment of the performance and thought that it is important since for most of the real life cases, execution time becomes distinctive.

In order to measure the execution time, we made use of a built-in function in R and included the results in the output tables.

4.1.3.4 Number of Iterations

Iteration number refers to the quantity that how many times the estimate is updated until convergence is achieved. It is an important characteristic in numerical studies since it is directly related with the time and store management e.g. less number of iterations saves both time and storage.

In this study, most of the numerical algorithms used have similar iteration processes so that it is considered as a comparable criterion. However, as briefly explained in Chapter 3, Nelder-Mead is a derivative-free simplex algorithm and it has a completely different working principle for iteration. As a result, its number of iterations are not comparable with the others. Nonetheless we keep it to compare its performance under different conditions. In addition, number of iterations illustrated in the tables are calculated by taking the average of the results obtained from 10000 runs.

4.1.3.5 MSE for Overall Fit

The comparison criteria in Section 4.1.3.1 and 4.1.3.2 are calculated for each parameter that is being estimated. On the other hand, another criterion, called as mean squared error for overall fit is needed to observe how well the estimated model fit.

Calculations for each fitted model is obtained by taking the average of the squared residuals. Residuals are the difference between the observed value of the dependent variable and its predicted value obtained from the estimated model fit and is given by

$$r_i = Y_i - \widehat{Y}_i \tag{4.11}$$

where Y_i denotes the actual value and $\widehat{Y}_i$ denotes the predicted value of the response variable.

Then, MSE for overall fit is obtained by

$$MSE(fit) = \frac{\sum_{i=1}^{n} {r_i}^2}{n}. \tag{4.12}$$

4.2 Results of the Simulation Study and Application

In this section, results of the data application and simulation study will be illustrated. The results are evaluated under all conditions one by one in order to discriminate the effect of each condition separately.

4.2.1 Simple Model Results

The results of all simulated scenarios will be given under 4 subsections according to distribution of errors, sample size, initial guesses and robustness. The model is the "Chwirut1" model as explain in Section 4.1.1.1.

Real Data Case

Before moving on with the simulation study results, real data results with comments are given. Real data results of Chwirut1 with good initial values are given in Table 4.1. As it can be observed from Table 4.1, except CG which is slightly higher, biases and MSE values for the estimates of the parameters are more or less the same and pretty small for all algorithms as desired. Another criterion is the MSE value for the fitted model and they are very close to each other for all methods except CG which is again not dramatically higher. When the execution time results are checked, the L-M and DFP methods are observed as the fastest ones with 0.04 seconds. However, L-M method converges with less number of iterations compared to DFP method as well as the other methods. To conclude, all methods provide reasonable estimates for the real data of the simple model when iterations are started with good initial values. To sum up, from the best to the worst, the algorithms can be ordered as L-M > DFP > Newton > BFGS > Nelder-Mead > CG.

Table 4.1: Real data results of Chwirut1 with good initial values

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	$3.37\text{x}10^{-4}$	**$2.77\text{x}10^{-4}$**	$-3.21\text{x}10^{-3}$	$2.64\text{x}10^{-4}$	-0.091	$2.78\text{x}10^{-4}$
	β_2	$1.32\text{x}10^{-4}$	**$1.31\text{x}10^{-4}$**	$1.93\text{x}10^{-4}$	$1.32\text{x}10^{-4}$	$-1.17\text{x}10^{-3}$	$1.31\text{x}10^{-4}$
	β_3	$2.84\text{x}10^{-5}$	**$3.09\text{x}10^{-5}$**	$9.49\text{x}10^{-6}$	$2.85\text{x}10^{-5}$	$3.41\text{x}10^{-3}$	$3.09\text{x}10^{-5}$
MSE	β_1	$7.74\text{x}10^{-5}$	**$2.21\text{x}10^{-4}$**	$4.84\text{x}10^{-5}$	$6.47\text{x}10^{-7}$	$8.13\text{x}10^{-3}$	$2.01\text{x}10^{-7}$
	β_2	$4.43\text{x}10^{-5}$	**$1.46\text{x}10^{-6}$**	$1.24\text{x}10^{-9}$	$1.42\text{x}10^{-6}$	$7.24\text{x}10^{-6}$	$5.41\text{x}10^{-8}$
	β_3	$3.18\text{x}10^{-6}$	**$4.51\text{x}10^{-8}$**	$6.38\text{x}10^{-9}$	$5.28\text{x}10^{-7}$	$1.16\text{x}10^{-5}$	$1.65\text{x}10^{-10}$
Time		0.1	**0.04**	0.1	0.08	0.09	0.04
Iterations		22	**8**	12	146	10	13
MSE (fit)		11.14	**11.14**	11.16	11.14	12.23	11.14

Table 4.2: Real data results of Chwirut1 with poor initial values

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	$5.16\text{x}10^{-4}$	**$2.77\text{x}10^{-4}$**	$-9.06\text{x}10^{-3}$	0.336	-0.091	$2.78\text{x}10^{-4}$
	β_2	$1.33\text{x}10^{-4}$	**$1.31\text{x}10^{-4}$**	$1.42\text{x}10^{-4}$	$2.78\text{x}10^{-3}$	$-1.18\text{x}10^{-3}$	$1.31\text{x}10^{-4}$
	β_3	$2.32\text{x}10^{-5}$	**$3.09\text{x}10^{-5}$**	$2.12\text{x}10^{-4}$	$-8.02\text{x}10^{-3}$	$3.41\text{x}10^{-3}$	$3.09\text{x}10^{-5}$
MSE	β_1	$1.04\text{x}10^{-7}$	**$1.84\text{x}10^{-4}$**	$1.28\text{x}10^{-5}$	0.142	$8.41\text{x}10^{-3}$	$4.77\text{x}10^{-7}$
	β_2	$2.42\text{x}10^{-6}$	**$3.62\text{x}10^{-5}$**	$2.25\text{x}10^{-8}$	$4.76\text{x}10^{-6}$	$1.54\text{x}10^{-6}$	$5.72\text{x}10^{-7}$
	β_3	$6.89\text{x}10^{-7}$	**$2.96\text{x}10^{-7}$**	$5.14\text{x}10^{-7}$	$4.51\text{x}10^{-5}$	$6.41\text{x}10^{-5}$	$5.57\text{x}10^{-9}$
Time		0.09	**0.05**	0.1	0.1	0.1	0.04
Iterations		25	**14**	32	144	10	17
MSE (fit)		11.14	**11.14**	11.18	16.43	12.23	11.14

The methods work well when iteration process is started with good initial values. The question is that if they can pursue their performance with the presence of poorly selected initial values. The results are presented in Table 4.2. When bias and MSE values are examined, it is seen that for all methods except Nelder-Mead and CG, they are close to each other. However, overall bias is ignorable even for Nelder-Mead and CG. Except for CG and Nelder-Mead which is the worst, other algorithms sustain their success in estimation with the same MSE for fit values.

The execution times are more or less the same as in the case of good initial values, i.e., there is no significant slowing down behavior. The number of iterations, on the other hand, increased for most of the methods compared to the results given in Table 4.1.

Although the number of iterations shows a drastic incerase in Nelder-Mead, it still provides a reasonably good fit. It can be concluded that all methods performed well with poor initial values as well as good initial values for this data set and the model.

Simulated Data Case

After presenting real data results, the comparisons through simulation are given for the simple model in the following sections.

4.2.1.1 Comparisons with respect to Error Distribution

As mentioned in Section 4.1.1.2, the algorithms are run for simulated data sets with both normal and non-normal error terms.

As non-normal distribution, generalized logistic distribution are selected with two different shape parameters. As a result, the algorithms are tested with both symmetric, right-skewed and left-skewed error terms. Since the real data for the model shows a right-skewed behavior, it is suspected that simulated data with right-skewed error may result in better fit compared to the left-skewed competitor.

By taking other conditions fixed, the effect of error distribution on the performance of the algorithms is considered. Firstly, consider the sample size is large and the iteration process is initiated with good starting values.

Table 4.3: Large sample + normal error + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-3.72×10^{-3}	2.12×10^{-5}	1.4×10^{-2}	-2.71×10^{-5}	9.01×10^{-2}	$\mathbf{-2.12\times10^{-4}}$
	β_2	-6.67×10^{-6}	7.28×10^{-8}	1.07×10^{-4}	-7.93×10^{-9}	1.51×10^{-4}	$\mathbf{-6.70\times10^{-7}}$
	β_3	7.55×10^{-5}	-3.03×10^{-7}	-3.66×10^{-4}	6.91×10^{-7}	2.02×10^{-3}	$\mathbf{5.05\times10^{-6}}$
MSE	β_1	3.56×10^{-4}	1.50×10^{-5}	1.33×10^{-2}	1.92×10^{-5}	8.12×10^{-3}	$\mathbf{1.41\times10^{-5}}$
	β_2	1.14×10^{-9}	1.51×10^{-10}	6.48×10^{-8}	1.62×10^{-10}	3.20×10^{-8}	$\mathbf{1.35\times10^{-10}}$
	β_3	1.44×10^{-7}	6.51×10^{-9}	2.52×10^{-6}	8.21×10^{-9}	4.39×10^{-6}	$\mathbf{6.09\times10^{-9}}$
Time		0.022	0.0025	0.0267	0.0297	0.0299	**0.0066**
Iterations		24.33	7.1	8.24	199.82	23.48	**10.36**
MSE (fit)		1.10	0.99	3.28	0.99	4.72	**0.98**

Table 4.4: Large sample + GL error (b=0.5) + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	3.55×10^{-2}	$\mathbf{3.71\times10^{-2}}$	5.07×10^{-2}	$\mathbf{3.7\times10^{-2}}$	-9.01×10^{-2}	3.50×10^{-2}
	β_2	6.18×10^{-5}	$\mathbf{6.47\times10^{-5}}$	1.59×10^{-4}	$\mathbf{6.47\times10^{-5}}$	-1.56×10^{-4}	6.69×10^{-5}
	β_3	-1.55×10^{-4}	$\mathbf{-1.88\times10^{-4}}$	-5.40×10^{-4}	$\mathbf{-1.8\times10^{-4}}$	2.69×10^{-3}	-1.96×10^{-4}
MSE	β_1	1.59×10^{-3}	$\mathbf{1.49\times10^{-3}}$	1.39×10^{-2}	$\mathbf{1.49\times10^{-3}}$	8.12×10^{-3}	1.52×10^{-3}
	β_2	5.38×10^{-9}	$\mathbf{5.24\times10^{-9}}$	5.85×10^{-8}	$\mathbf{5.25\times10^{-9}}$	3.44×10^{-8}	5.68×10^{-9}
	β_3	1.55×10^{-7}	$\mathbf{8.27\times10^{-8}}$	2.22×10^{-6}	$\mathbf{8.45\times10^{-8}}$	7.58×10^{-6}	8.77×10^{-8}
Time		0.025	**0.0025**	0.0267	**0.0274**	0.0343	0.0068
Iterations		24.78	**7.02**	8.09	**188.7**	21.06	10.78
MSE (fit)		6.60	**6.54**	7.99	**6.54**	12.61	6.58

Table 4.5: Large sample + GL error (b=2) + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-3.01×10^{-2}	$\mathbf{-2.45\times10^{-2}}$	-1.27×10^{-2}	-2.47×10^{-2}	-9.01×10^{-2}	$\mathbf{-2.50\times10^{-2}}$
	β_2	-5.48×10^{-5}	$\mathbf{-4.47\times10^{-7}}$	6.89×10^{-5}	-4.50×10^{-5}	1.56×10^{-4}	$\mathbf{-4.59\times10^{-5}}$
	β_3	2.25×10^{-4}	$\mathbf{1.09\times10^{-4}}$	-2.24×10^{-4}	1.13×10^{-4}	1.66×10^{-3}	$\mathbf{1.15\times10^{-4}}$
MSE	β_1	1.29×10^{-3}	$\mathbf{6.31\times10^{-4}}$	1.26×10^{-2}	6.49×10^{-4}	8.12×10^{-3}	$\mathbf{6.41\times10^{-4}}$
	β_2	4.36×10^{-9}	$\mathbf{2.32\times10^{-9}}$	4.89×10^{-8}	2.38×10^{-9}	3.26×10^{-8}	$\mathbf{2.45\times10^{-9}}$
	β_3	2.05×10^{-7}	$\mathbf{2.61\times10^{-8}}$	2.41×10^{-6}	3.10×10^{-8}	2.97×10^{-6}	$\mathbf{2.78\times10^{-8}}$
Time		0.0291	**0.0029**	0.0345	0.0268	0.0407	**0.0074**
Iterations		23.39	**7.57**	8.36	193.32	22.39	**10.24**
MSE (fit)		2.41	**2.28**	4.78	2.28	4.81	**2.27**

The results are presented in Tables 4.3, 4.4 and 4.5, respectively. The estimates with good statistical properties are obtained when the errors are normally distributed, which is expected. Under the non-normality assumption of errors, the results are plausible. When the error distribution is right skewed, the estimates seem more precise compared to the left skewed one. L-M, DFP and Nelder-Mead methods provided very close fits, separately. Besides, Newton method competes with them. However, L-M stands out with shortest execution time and least number of iterations, which makes it the best for these cases.

The fits that BFGS method provided is generally on average looking at the presented tables. However, BFGS method deals with the left-skewed errors more successfully compared to its closest competitor conjugate gradient method. So, it can be concluded that the conjugate gradient method gives the worst fit with the presence of any type of error distribution.

To wrap things up, the most successful one under all these conditions seems to be L-M method because it provides one of the best fits with the less number of iterations and within the shortest amount of time. If the algorithms are ordered from the best to the worst for this specific case, it should be as L-M > Nelder-Mead > DFP > Newton > BFGS > CG.

We have interpreted the results obtained only when the good initials are assigned at the beginning of the estimation process. The results obtained when the poor initials are assigned under the same conditions are not given here since they are very similar to good initials case.

4.2.1.2 Comparisons with respect to Sample Size

Sample size is another condition that is the interest of this comparative study as mentioned in Section 4.1.1.4. It is classified as small and large sample size for this study. The large sample size is determined as 250 while the small sample size is determined as 25. The difference between them is significant since we want to examine the behavior of algorithms under such situation. The aim here is to observe which one(s) of them handles this situation better compared to its competitors when other conditions are hold fixed, i.e., normally distributed errors and good initial values. The simulation results are as presented in Table 4.3 and Table 4.6, for small and large sample sizes. respectively.

Table 4.6: Small sample + normal error + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-4.20×10^{-3}	$\mathbf{1.05\times10^{-4}}$	3.01×10^{-2}	-3.37×10^{-4}	-9.01×10^{-2}	6.17×10^{-6}
	β_2	-1.01×10^{-5}	$\mathbf{-2.24\times10^{-7}}$	1.70×10^{-4}	-2.76×10^{-6}	-2.44×10^{-4}	6.68×10^{-7}
	β_3	9.14×10^{-5}	$\mathbf{9.26\times10^{-7}}$	-7.58×10^{-4}	1.32×10^{-5}	2.39×10^{-3}	1.52×10^{-6}
MSE	β_1	5.87×10^{-4}	$\mathbf{1.83\times10^{-4}}$	6.80×10^{-3}	2.22×10^{-4}	8.12×10^{-3}	1.78×10^{-4}
	β_2	6.81×10^{-9}	$\mathbf{4.35\times10^{-9}}$	7.91×10^{-8}	5.66×10^{-9}	1.01×10^{-7}	6.51×10^{-9}
	β_3	2.70×10^{-7}	$\mathbf{9.35\times10^{-8}}$	2.21×10^{-6}	1.20×10^{-7}	6.38×10^{-6}	9.47×10^{-8}
Time		0.0195	**0.0015**	0.02	0.0207	0.0211	0.0041
Iterations		22.84	**7.39**	9.57	194.78	19.76	10.73
MSE (fit)		0.98	**0.88**	2.28	0.89	5.38	0.89

According to the results presented in Table 4.3 and Table 4.6, all methods perform slightly better overall with smaller sample size under normally distributed errors and good initial values conditions. Under such perfect conditions, this result may be attributed to simulation error other than inconsistency. However, when the MSE fit values are examined, it is crystal clear that L-M, Nelder-Mead and DFP methods perform as the best and Newton-Raphson follows them. To sum up, they all perform well under both situations and could easily handle the small sample size. They even provide improved estimates with a slightly shorter amount of execution time with small sample size.

Another comparison for sample size will be made with the simulated datasets when error term is generated from generalized logistic distribution with a shape parameter b=2. The initial values are assigned as close to the certified values, in other words good initial values.

Table 4.7: Small sample + GL (b=2) error + good initial values (Chwirut1)

		Newton	**L-M**	**BFGS**	**Nelder-Mead**	**CG**	**DFP**
Bias	β_1	-3.13×10^{-2}	$\mathbf{-2.49\times10^{-2}}$	5.31×10^{-3}	-2.52×10^{-2}	-9.01×10^{-2}	-2.50×10^{-2}
	β_2	-6.74×10^{-5}	$\mathbf{-5.14\times10^{-5}}$	1.28×10^{-4}	-5.41×10^{-5}	-2.33×10^{-4}	-5.07×10^{-5}
	β_3	2.71×10^{-4}	$\mathbf{1.33\times10^{-4}}$	-6.64×10^{-4}	1.46×10^{-4}	-1.93×10^{-3}	1.32×10^{-4}
MSE	β_1	1.73×10^{-3}	$\mathbf{9.78\times10^{-4}}$	4.09×10^{-3}	1.03×10^{-3}	8.12×10^{-3}	9.72×10^{-4}
	β_2	1.65×10^{-8}	$\mathbf{1.15\times10^{-8}}$	5.59×10^{-8}	1.28×10^{-8}	8.49×10^{-8}	9.94×10^{-9}
	β_3	4.50×10^{-7}	$\mathbf{2.09\times10^{-7}}$	1.68×10^{-6}	2.34×10^{-7}	4.38×10^{-6}	2.01×10^{-7}
Time		0.0232	**0.0019**	0.0248	0.0246	0.0252	0.0039
Iterations		21.58	**7.52**	9.73	188.48	20.31	10.34
MSE (fit)		2.14	**2.02**	3.64	2.02	5.42	2.06

According to the results presented in Table 4.5 and Table 4.7, same conclusions for the previous comparison scenario apply to this situation as well. Again there is a slight improvement in the estimates when the sample size is smaller. Based on bias and MSE

values of the estimates and MSE for overall fit, it can be concluded that the best ones still are the L-M, Nelder-Mead and DFP methods. Besides these 3 methods, Newton's method performed quite well, too. Lastly, we will present the results for comparison when the error has a generalized logistic distribution with the shape parameter b=0.5 and the initial values are poorly selected. Now, the presence of poor initials makes the problem a little bit complex. Tables 4.8 and 4.9 show the corresponding results regarding the both sample sizes, respectively.

Table 4.8: Large sample + GL (b=0.5) error + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	0.14	**0.036**	0.056	0.252	5.193	0.047
	β_2	5.92×10^{-5}	$\mathbf{6.37\times10^{-5}}$	1.18×10^{-4}	2.76×10^{-4}	-0.858	1.69×10^{-5}
	β_3	-8.65×10^{-4}	$\mathbf{-1.80\times10^{-4}}$	2.54×10^{-3}	-3.01×10^{-3}	-0.138	1.87×10^{-3}
MSE	β_1	0.192	$\mathbf{1.47\times10^{-3}}$	0.041	0.114	3906.9	8.48×10^{-3}
	β_2	2.07×10^{-7}	$\mathbf{5.02\times10^{-9}}$	2.45×10^{-7}	5.77×10^{-7}	228.4	5.39×10^{-7}
	β_3	2.78×10^{-5}	$\mathbf{7.92\times10^{-8}}$	4.38×10^{-4}	6.56×10^{-5}	19.99	3.55×10^{-4}
Time		0.0223	**0.0019**	0.0236	0.0218	0.0228	0.0125
Iterations		35.19	**11.06**	20.85	229.9	47.61	14.71
MSE (fit)		17.75	**6.585**	60.67	22.52	1926	41.58

Table 4.9: Small sample + GL (b=0.5) error + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	0.095	**0.039**	0.071	0.228	4.448	0.047
	β_2	-1.17×10^{-3}	$\mathbf{8.05\times10^{-5}}$	-5.41×10^{-4}	-1.25×10^{-4}	-1.005	1.34×10^{-4}
	β_3	4.05×10^{-3}	$\mathbf{-2.48\times10^{-4}}$	2.67×10^{-3}	-1.95×10^{-3}	-0.123	1.94×10^{-4}
MSE	β_1	0.125	$\mathbf{2.99\times10^{-3}}$	0.049	0.099	2283.7	9.93×10^{-3}
	β_2	9.48×10^{-5}	$\mathbf{3.91\times10^{-8}}$	5.04×10^{-5}	4.07×10^{-5}	57.11	1.19×10^{-6}
	β_3	1.29×10^{-3}	$\mathbf{7.54\times10^{-7}}$	7.01×10^{-3}	3.06×10^{-4}	3.525	4.79×10^{-5}
Time		0.0837	**0.0011**	0.0848	0.0836	0.0838	0.0047
Iterations		30.28	**11.67**	21.84	224.4	41.83	11.15
MSE (fit)		58.91	**5.774**	100.3	21.71	1590	29.64

The results imply that CG and DFP methods present worse outcomes when the sample size decreases dramatically. Others almost satisfy consistency. They could not handle both poor initials, negatively skewed error distribution and small sample size, simultaneously. On the other hand, looking at the MSE for fit values, L-M, Nelder-Mead and DFP methods provide better fits when the sample size is smaller. Although CG method performs better for small samples as compared to all others it is the worst one in terms of MSE for the fit. Overall, L-M method is clearly the best one among them to handle the whole criteria represented in this section.

Comparison based on small and large sample sizes was the subject of this section and results are interpreted with respect to both precision and consistency features. As a final result, L-M seems to take the first place among the others.

4.2.1.3 Comparisons with respect to Initial Values

Initial value specification is a very important subject in any type of optimization problem as explained briefly in Section 4.1.1.3. In our study, we define 2 sets of starting values to initiate the estimation process. One of them is reasonably close to the true solution which is called as certified values in our case. The other set was selected as far from the true solution so that it is classified as poor initial values.

In this section of the study, the simulation results obtained for the simple model will be shown and examined with both good and poor initial values under different conditions.

The first comparison is carried out on the simulated dataset whose error term is normally distributed and sample size is large. The results obtained by starting the process with both good and poor initial values are compared and given in Tables 4.3 and 4.10, respectively. In fact, normally distributed errors and large sample size case can be considered as ideal so successful results can be expected from all algorithms.

Looking at the presented results, when good initials were replaced by poor initials, all methods except L-M method resulted in worse estimates with higher bias and MSE values, respectively. Only L-M method preserved its performance in estimation.

When we look at the Table 4.3, we see that L-M method, Nelder-Mead method and DFP method provided the best fits with very small and close MSE fit values. In addition to them, Newton's method followed them with a very close MSE for overall fit value.

Table 4.10: Large sample + normal error + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	0.094	**5.69×10^{-5}**	0.015	0.237	6.078	0.011
	β_2	1.92×10^{-5}	**1.18×10^{-7}**	3.45×10^{-5}	2.59×10^{-4}	-3.894	-6.34×10^{-5}
	β_3	-7.99×10^{-4}	**-1.26×10^{-6}**	-3.16×10^{-3}	-3.29×10^{-3}	-1.415	2.26×10^{-3}
MSE	β_1	0.169	**1.51×10^{-5}**	0.019	0.114	15334.4	6.89×10^{-3}
	β_2	3.71×10^{-8}	**1.56×10^{-10}**	3.10×10^{-7}	4.84×10^{-7}	10903.2	3.86×10^{-7}
	β_3	9.75×10^{-6}	**6.56×10^{-9}**	5.01×10^{-4}	5.31×10^{-5}	1724.5	4.16×10^{-4}
Time		0.0206	**0.0018**	0.028	0.0192	0.032	0.0127
Iterations		34.63	**11.18**	20.74	232.7	45.03	15.24
MSE (fit)		9.315	**0.991**	60.29	17.63	2071	35.12

If the results in the Table 4.10 are checked, it can be seen that L-M method still is the best while Newton-Raphson method follows it. Nelder-Mead method and DFP method get worse when poor initial values are assigned in the beginning of the algorithm. Moreover, BFGS method performs very poorly compared to the results with good initial values. Lastly, conjugate gradient method is the worst for this case again since it provides an unacceptable fit with biased estimates. Furthermore, if number of iterations and execution time are checked in both tables, it can be observed that there is an increase in the presence of poor initial values, which is not unexpected.

Another comparison between good and poor initial values is applied on the simulated dataset whose error term is distributed as generalized logistic with shape parameter b=2 and sample size is large and given in Table 4.5 and Table 4.11, respectively.

Table 4.11: Large sample + GL (b=2) error + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	0.078	**-0.024**	-8.39×10^{-3}	0.228	7.623	-8.20×10^{-3}
	β_2	-3.57×10^{-5}	$\mathbf{-4.45\times10^{-5}}$	-2.63×10^{-5}	2.69×10^{-4}	-0.795	-7.14×10^{-5}
	β_3	-5.71×10^{-4}	$\mathbf{1.07\times10^{-4}}$	4.51×10^{-3}	-3.68×10^{-3}	-0.192	2.30×10^{-3}
MSE	β_1	0.167	$\mathbf{6.24\times10^{-4}}$	0.029	0.107	7303.3	0.011
	β_2	1.98×10^{-7}	$\mathbf{2.29\times10^{-9}}$	2.98×10^{-7}	1.56×10^{-7}	73.79	7.02×10^{-7}
	β_3	5.03×10^{-5}	$\mathbf{2.54\times10^{-8}}$	7.08×10^{-4}	2.46×10^{-5}	17.47	4.16×10^{-4}
Time		0.0231	**0.0022**	0.0291	0.0208	0.0283	0.0142
Iterations		34.64	**11.56**	20.89	232.3	44.18	15.77
MSE (fit)		14.06	**2.277**	80.07	17.07	2172	42.65

When the results with good initial values are checked, it is clear that estimates are plausible for almost all algorithms. In order, DFP method, L-M method, Nelder-Mead method, Newton method provided good solution to the problem. On the other hand, BFGS method and conjugate gradient method cannot be counted as very successful beside them.

When poor initial values step in, L-M method becomes the best algorithm for the problem by far. Biases and MSE values of the others get higher especially for BFGS and conjugate gradient methods. Since conjugate gradient method failed drastically for almost all cases when the model is simple, it is not woth to consider it for the complex model.

Lastly, the comparisons based on the simulated data set with small sample size, left-skewed error distribution which is GL(b=0.5) and poor and good initial values are summarized in Table 4.9 and Table 4.12, respectively.

Table 4.12: Small sample + GL (b=0.5) error + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	$3.78\text{x}10^{-2}$	**$4.01\text{x}10^{-2}$**	$5.83\text{x}10^{-3}$	$3.94\text{x}10^{-2}$	$-9.01\text{x}10^{-2}$	$3.97\text{x}10^{-2}$
	β_2	$7.01\text{x}10^{-5}$	**$7.48\text{x}10^{-5}$**	$2.20\text{x}10^{-4}$	$7.19\text{x}10^{-5}$	$-2.57\text{x}10^{-4}$	$7.54\text{x}10^{-5}$
	β_3	$-1.92\text{x}10^{-4}$	**$-2.37\text{x}10^{-4}$**	$-7.95\text{x}10^{-4}$	$-2.23\text{x}10^{-4}$	$3.02\text{x}10^{-3}$	$-2.22\text{x}10^{-4}$
MSE	β_1	$3.19\text{x}10^{-3}$	**$3.01\text{x}10^{-3}$**	$7.42\text{x}10^{-3}$	$3.03\text{x}10^{-3}$	$8.12\text{x}10^{-3}$	$2.99\text{x}10^{-3}$
	β_2	$3.31\text{x}10^{-8}$	**$3.19\text{x}10^{-8}$**	$8.87\text{x}10^{-8}$	$3.38\text{x}10^{-8}$	$1.41\text{x}10^{-7}$	$3.28\text{x}10^{-8}$
	β_3	$8.44\text{x}10^{-7}$	**$7.14\text{x}10^{-7}$**	$2.01\text{x}10^{-6}$	$7.51\text{x}10^{-7}$	$1.01\text{x}10^{-5}$	$7.03\text{x}10^{-7}$
Time		0.0191	**0.0014**	0.0219	0.0204	0.0203	0.0038
Iterations		23.46	**7.39**	8.1	193.49	19.99	10.99
MSE (fit)		5.92	**5.86**	7.26	5.87	12.63	5.75

L-M, Nelder-Mead, DFP and Newton-Raphson methods provide very close MSE for overall fit value when iterations start with good initials. However, only L-M method becomes robust under poor initials. Nelder-Mead and DFP method follow it in order when MSE for overall fit is taken into consideration, but there is a significant difference between the estimated fits. Other than that, the others could not suggest reasonable fits. More importantly, conjugate gradient method failed to converge seriously.

To conclude, only L-M method could handle the iteration process starting with set of poor initial values. In addition to that, it achieves convergence with reasonably less number of iterations and within the shortest amount of time.

4.2.1.4 Comparisons with respect to Robustness

Finally, the iterative methods are compared with respect to their robustness to abnormalities in the error term. The comparison will be made under 3 subclasses which are robustness to outliers, robustness to contamination and robustness to inliers in error term.

Comparisons with respect to Robustness to Outliers in Errors

In this section, by assuming 10% of the errors comes from N(0,4), the performance of the algorithms is evaluated. The first comparison will be carried out for the ideal case whose results are illustrated in Table 4.3 and its version with outliers is given in Table 4.13.

Table 4.13: Large sample + error with outliers + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-4.39×10^{-3}	3.43×10^{-4}	2.14×10^{-2}	3.52×10^{-4}	-9.01×10^{-2}	**4.46×10^{-4}**
	β_2	-8.93×10^{-6}	-5.72×10^{-8}	1.05×10^{-4}	-5.58×10^{-8}	-1.67×10^{-4}	**8.47×10^{-7}**
	β_3	9.53×10^{-5}	-9.65×10^{-7}	-4.41×10^{-4}	-1.13×10^{-6}	2.08×10^{-3}	**-6.91×10^{-6}**
MSE	β_1	4.27×10^{-4}	1.92×10^{-5}	1.59×10^{-2}	1.93×10^{-5}	8.12×10^{-3}	**1.98×10^{-5}**
	β_2	1.42×10^{-9}	1.90×10^{-10}	3.03×10^{-8}	1.89×10^{-10}	3.39×10^{-8}	**1.96×10^{-10}**
	β_3	1.76×10^{-7}	7.75×10^{-9}	2.96×10^{-6}	7.81×10^{-9}	4.53×10^{-6}	**9.66×10^{-9}**
Time		0.026	0.002	0.028	0.027	0.033	**0.0045**
Iterations		24.26	7.04	8.63	195.38	23.35	**10.35**
MSE (fit)		1.45	1.31	2.99	1.31	4.87	**1.27**

As it can be clearly concluded by comparing the results in both tables, ranking with respect to overall performance does not change when the outliers are added to the error term. MSE for the fit values slightly increased, but it is not that significant. To sum up, all methods provide nearly perfect fits with or without outliers. However, this conclusion is only for the ideal case. Under worse situations, the results may be completely different. To prove this hypothesis, robustness of the algorithms against outliers in the error distribution is checked for poor initial values and the results are summarized in Table 4.10 and Table 4.14 .

Table 4.14: Large sample + error with outliers + poor initial values (Chwirut1)

		Newton	**L-M**	**BFGS**	**Nelder-Mead**	**CG**	**DFP**
Bias	β_1	$3.24\text{x}10^{-3}$	$1.73\text{x}10^{-3}$	0.165	0.597	2.413	**$1.65\text{x}10^{-2}$**
	β_2	$-1.09\text{x}10^{-3}$	$6.87\text{x}10^{-4}$	$2.48\text{x}10^{-5}$	$-4.45\text{x}10^{-4}$	-0.223	**$1.21\text{x}10^{-5}$**
	β_3	$1.29\text{x}10^{-2}$	$-6.75\text{x}10^{-4}$	$2.06\text{x}10^{-2}$	$7.94\text{x}10^{-3}$	0.303	**$-2.09\text{x}10^{-4}$**
MSE	β_1	$6.01\text{x}10^{-2}$	$1.22\text{x}10^{-3}$	0.195	0.904	777.64	**$1.36\text{x}10^{-2}$**
	β_2	$3.47\text{x}10^{-5}$	$1.18\text{x}10^{-4}$	$5.95\text{x}10^{-7}$	$1.47\text{x}10^{-5}$	12.09	**$9.65\text{x}10^{-9}$**
	β_3	$4.54\text{x}10^{-3}$	$1.21\text{x}10^{-4}$	$6.90\text{x}10^{-3}$	$5.21\text{x}10^{-3}$	1.049	**$2.21\text{x}10^{-6}$**
Time		0.028	0.006	0.044	0.026	0.0328	**0.008**
Iterations		38.04	15.02	24.18	244.8	39.17	**13.67**
MSE (fit)		126.78	6.05	262.38	175.39	3197.8	**2.23**

When the outcomes are examined, it can be seen that there is a significant difference between the results of two cases. In the regular case, L-M method offers the best fit which is almost perfect. After that, Newton-Raphson and Nelder-Mead methods follows it with nearly plausible fits. On the contrary, conjugate gradient method fails to converge and could not produce an acceptable fit. When outliers are added to the error term, it is seen that L-M method still produces a good fit, but DFP method takes over the first place in such case. It has better bias and MSE results than Levenberg method.

Besides, Newton, BFGS and Nelder-Mead methods seem to have some problems in converging looking at their MSE for overall fit values, which are high. Furthermore, conjugate gradient is the worst one as usual. It certainly failed to converge.

Comparisons with respect to Robustness to Contamination in Errors

As discussed earlier, we contaminated the error distribution, which is standard normal distribution, with standard uniform distribution to test the robustness of the algorithms to contamination.

The simulation results obtained under the assumption of contaminated normal error terms for both small and large sample sizes and good and poor intials are given as follows.

Table 4.15: Large sample + contaminated error + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-2.88×10^{-3}	$\mathbf{-1.13\times10^{-3}}$	-1.93×10^{-3}	-1.12×10^{-3}	-9.01×10^{-2}	-1.53×10^{-3}
	β_2	-6.56×10^{-6}	$\mathbf{-2.68\times10^{-6}}$	7.31×10^{-5}	-2.64×10^{-6}	-1.45×10^{-4}	-3.16×10^{-6}
	β_3	4.02×10^{-5}	$\mathbf{3.81\times10^{-6}}$	-1.47×10^{-4}	3.55×10^{-6}	1.97×10^{-3}	1.32×10^{-5}
MSE	β_1	1.76×10^{-4}	$\mathbf{1.24\times10^{-5}}$	5.63×10^{-4}	1.25×10^{-5}	8.12×10^{-3}	1.62×10^{-5}
	β_2	7.83×10^{-10}	$\mathbf{1.11\times10^{-10}}$	9.49×10^{-9}	1.10×10^{-10}	3.07×10^{-8}	1.18×10^{-10}
	β_3	7.29×10^{-8}	$\mathbf{5.17\times10^{-9}}$	2.26×10^{-7}	5.19×10^{-9}	4.13×10^{-6}	5.95×10^{-9}
Time		0.025	**0.002**	0.0256	0.0264	0.0261	0.0048
Iterations		25.03	**7.12**	8.25	199.18	22.86	10.38
MSE (fit)		0.97	**0.92**	1.41	0.92	4.39	0.92

Table 4.16: Small sample + contaminated error + good initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-5.82×10^{-3}	-2.01×10^{-3}	2.02×10^{-2}	-2.01×10^{-3}	-9.01×10^{-2}	$\mathbf{-1.73\times10^{-3}}$
	β_2	-1.42×10^{-5}	-4.32×10^{-6}	1.51×10^{-4}	-4.31×10^{-6}	-2.79×10^{-4}	$\mathbf{-5.91\times10^{-7}}$
	β_3	9.31×10^{-5}	1.05×10^{-5}	-6.39×10^{-4}	1.05×10^{-5}	2.45×10^{-3}	$\mathbf{2.44\times10^{-7}}$
MSE	β_1	5.55×10^{-4}	1.54×10^{-4}	2.35×10^{-3}	1.54×10^{-4}	8.12×10^{-3}	$\mathbf{1.46\times10^{-4}}$
	β_2	6.23×10^{-9}	4.32×10^{-9}	3.67×10^{-8}	4.31×10^{-9}	1.46×10^{-7}	$\mathbf{3.31\times10^{-9}}$
	β_3	2.51×10^{-7}	8.17×10^{-8}	1.15×10^{-6}	8.17×10^{-8}	7.02×10^{-6}	$\mathbf{7.41\times10^{-8}}$
Time		0.0231	0.001	0.0227	0.0221	0.0228	**0.0018**
Iterations		23.03	7.41	9.33	192.98	17.68	**10.66**
MSE (fit)		0.89	0.82	1.97	0.82	5.31	**0.81**

Table 4.17: Large sample + contaminated error + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	8.75×10^{-2}	$\mathbf{-7.85\times10^{-4}}$	8.39×10^{-2}	0.577	28.975	4.59×10^{-2}
	β_2	-2.28×10^{-3}	$\mathbf{-1.08\times10^{-6}}$	5.31×10^{-5}	-5.51×10^{-4}	-13.293	-1.21×10^{-4}
	β_3	1.85×10^{-2}	$\mathbf{-5.21\times10^{-6}}$	8.57×10^{-3}	8.13×10^{-3}	-6.233	5.93×10^{-3}
MSE	β_1	0.283	$\mathbf{1.31\times10^{-5}}$	8.34×10^{-2}	0.841	80700.9	3.99×10^{-2}
	β_2	1.76×10^{-4}	$\mathbf{1.38\times10^{-10}}$	4.82×10^{-7}	1.14×10^{-5}	17835.16	5.05×10^{-7}
	β_3	7.25×10^{-3}	$\mathbf{5.66\times10^{-9}}$	3.31×10^{-3}	3.73×10^{-3}	4388.9	2.51×10^{-3}
Time		0.0281	**0.002**	0.0358	0.0256	0.0336	0.0103
Iterations		37.92	**14.91**	24.59	255.19	39.28	13.16
MSE (fit)		198.14	**0.91**	116.88	220.52	3172.1	87.28

Table 4.18: Small sample + contaminated error + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	-6.21x10^{-2}	**6.35x10^{-3}**	0.114	0.936	1.115	6.93x10^{-2}
	β_2	-2.30x10^{-2}	**2.24x10^{-5}**	-8.67x10^{-3}	-3.67x10^{-3}	-0.877	2.38x10^{-5}
	β_3	7.55x10^{-2}	**-1.87x10^{-4}**	4.44x10^{-2}	9.72x10^{-3}	0.228	4.09x10^{-3}
MSE	β_1	0.213	**4.82x10^{-2}**	0.118	1.515	17.72	5.86x10^{-2}
	β_2	5.25x10^{-3}	**2.25x10^{-3}**	8.14x10^{-4}	8.97x10^{-4}	72.66	2.26x10^{-7}
	β_3	5.28x10^{-2}	**6.77x10^{-3}**	1.33x10^{-2}	1.38x10^{-2}	1.339	2.42x10^{-3}
Time		0.0235	**0.0012**	0.0236	0.0233	0.0264	0.0056
Iterations		32.41	**26.27**	25.62	269.75	27.13	16.51
MSE (fit)		450.37	**1.14**	464.38	290.51	2437.8	19.34

The Tables 4.15 and 4.16 illustrate the results of the cases with good initial values while Tables 4.17 and 4.18 covers the ones with poor initial values. When the first two tables are examined and compared with their versions with no contamination, which corresponds to the Tables 4.3 and 4.6, it is clearly seen that methods are not affected by the presence of contamination in errors. Even they produced better fits when the MSE for overall fit is checked. The execution time and the number of iterations do not change significantly, too. Thus, it can be concluded that contamination does not influence the performance of algorithms drastically when the iteration process is started with plausible initial values. In other words, robustness in such situations is achieved.

On the contrary, the results of the cases with poorly selected initial values are not satisfying for most of the algorithms. In order to make reliable comments, Tables 4.17 and 4.18 are examined. It is very clear that only Levenberg-Marqaurdt method gives a good fit. The other algorithms are not successful when contamination in errors and poor initial values are present simultaneously.

Comparisons with respect to Robustness to Inliers in Errors

Lastly, robustness of algorithms to inliers is tested. We will only present the outputs of the case when the initial values are poor because the results are not affected by inliers when the initial values are good. Tables 4.19 and 4.20 demonstrates the cases.

Table 4.19: Large sample + error with inliers + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	0.113	**3.51×10^{-5}**	0.109	0.541	0.717	2.48×10^{-2}
	β_2	-7.39×10^{-4}	**-1.25×10^{-6}**	-6.63×10^{-5}	-8.28×10^{-5}	-4.18×10^{-3}	-2.38×10^{-5}
	β_3	8.26×10^{-3}	**2.77×10^{-6}**	1.22×10^{-2}	2.10×10^{-3}	0.357	4.92×10^{-4}
MSE	β_1	0.289	**8.38×10^{-6}**	0.110	0.728	0.515	2.01×10^{-2}
	β_2	3.02×10^{-5}	**8.04×10^{-11}**	3.06×10^{-6}	7.17×10^{-6}	2.94×10^{-5}	1.48×10^{-7}
	β_3	3.23×10^{-3}	**3.26×10^{-9}**	3.55×10^{-3}	2.63×10^{-3}	0.131	5.33×10^{-5}
Time		0.053	**0.003**	0.029	0.024	0.0315	0.005
Iterations		38.69	**14.96**	24.44	250.22	36.72	13.19
MSE (fit)		104.69	**1.54**	169.56	129.89	3108.9	21.19

Table 4.20: Small sample + error with inliers + poor initial values (Chwirut1)

		Newton	L-M	BFGS	Nelder-Mead	CG	DFP
Bias	β_1	0.112	**3.27×10^{-2}**	0.144	1.035	0.702	8.25×10^{-2}
	β_2	-2.29×10^{-2}	**2.36×10^{-2}**	-5.09×10^{-3}	-5.26×10^{-3}	-3.01×10^{-2}	5.55×10^{-5}
	β_3	7.51×10^{-2}	**-4.31×10^{-2}**	2.51×10^{-2}	1.61×10^{-2}	0.334	-1.01×10^{-3}
MSE	β_1	5.706	**6.72×10^{-2}**	0.206	1.891	0.496	6.75×10^{-2}
	β_2	5.25×10^{-3}	**3.14×10^{-3}**	8.24×10^{-4}	1.32×10^{-3}	1.76×10^{-3}	7.21×10^{-8}
	β_3	5.06×10^{-2}	**1.27×10^{-2}**	4.62×10^{-2}	2.35×10^{-2}	0.119	1.06×10^{-5}
Time		0.021	**0.002**	0.021	0.020	0.027	0.003
Iterations		33.07	**23.67**	27.34	274.33	33.81	15.68
MSE (fit)		530.67	**5.68**	354.57	193.07	2508.3	8.73

Under such conditions, L-M gives the best fit while conjugate gradient method gives the worst one. Biases and MSE values are plausibly small for L-M and DFP methods. The other methods cannot provide unbiased estimates with low MSE values for this case. Since the gap between the provided fits are huge, there is no need to check other comparison criteria in order to decide on the best one, which is clearly L-M method.

4.2.2 Complex Model Results

In this section of the study, simulation study conducted for the complex model, namely "Thurber", will be examined. Before the simulation study, real data solutions is given with brief comments.

Real Data Case

Table 4.21: Real data results of Thurber with good initial values

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	1.421	**0.139**	-0.081	40.579	0.628
	β_2	-101.96	**0.085**	-17.031	-79.677	-55.527
	β_3	-74.977	**0.242**	-12.0	-63.177	-40.734
	β_4	-14.232	**0.417**	-1.998	-11.309	-7.626
	β_5	-0.083	**-3.70x10^{-3}**	-0.015	-0.101	-0.046
	β_6	-0.040	**-2.02x10^{-3}**	-8.09x10^{-3}	-0.013	-0.023
	β_7	-0.014	**-2.71x10^{-4}**	-3.51x10^{-3}	-5.57x10^{-3}	-8.41x10^{-3}
MSE	β_1	1.945	**0.021**	3.71x10^{-3}	1564.74	0.352
	β_2	989.51	**3.36x10^{-3}**	278.45	5978.26	3452.61
	β_3	4879.384	**0.109**	158.01	4012.34	1751.32
	β_4	241.84	**0.193**	3.568	148.91	61.13
	β_5	1.88x10^{-3}	**2.15x10^{-4}**	3.92x10^{-4}	0.025	2.91x10^{-4}
	β_6	6.92x10^{-4}	**5.81x10^{-6}**	7.41x10^{-4}	8.81x10^{-4}	3.52x10^{-4}
	β_7	2.54x10^{-4}	**6.87x10^{-7}**	3.14x10^{-5}	4.12x10^{-5}	8.75x10^{-5}
Time		0.01	**0.06**	0.41	0.04	0.02
Iterations		116	**25**	46	501	53
MSE (fit)		171.79	**152.505**	154.097	537.107	161.091

Table 4.22: Real data results of Thurber with poor initial values

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-13.171	**0.139**	0.957	-363.149	0.139
	β_2	-414.951	**0.074**	-10.344	-518.602	0.079
	β_3	-313.249	**0.234**	-7.607	-223.747	0.238
	β_4	-56.454	**0.415**	-1.164	-34.175	0.416
	β_5	-0.427	**-3.71x10^{-3}**	-0.011	-0.071	-3.70x10^{-3}
	β_6	-0.218	**-2.03x10^{-3}**	-5.42x10^{-3}	-0.212	-2.02x10^{-3}
	β_7	8.87x10^{-3}	**-2.73x10^{-4}**	-2.19x10^{-3}	-0.115	-2.72x10^{-4}
MSE	β_1	107.84	**0.024**	0.857	1478269	0.023
	β_2	21124.5	**5.87x10^{-3}**	112.3	5762458	7.78x10^{-4}
	β_3	84571.2	**0.043**	86.54	45988.1	1.15x10^{-3}
	β_4	2945.11	**0.108**	1.223	1245.35	0.106
	β_5	0.195	**2.13x10^{-4}**	1.59x10^{-4}	3.33x10^{-3}	2.01x10^{-5}
	β_6	0.071	**5.23x10^{-5}**	3.23x10^{-6}	0.069	4.99x10^{-6}
	β_7	8.11x10^{-5}	**8.14x10^{-7}**	5.89x10^{-6}	0.012	6.95x10^{-7}
Time		0.01	**0.02**	0.05	0.05	0.03
Iterations		113	**13**	55	502	87
MSE (fit)		2461.061	**152.505**	153.394	37650.81	152.506

The comparisons with real data under the assumption of good and poor initial values are given in Table 4.21 and Table 4.22, respectively. It is clear that none of the methods could provide a reasonable fit. That is the reason why this model and data is classified as “high level of difficulty” in the website. The results are not good compared to the results of the simple model. Since the number of parameters and complexity of the model increased, the methods struggle.

In Table 4.21, the process is started with the set of good initial values and L-M provides the best solution seemingly while the quasi-Newton methods follow it. On the other hand, Nelder-Mead method fails because its MSE for overall fit is extremely big.

The output with poor initial values are as given in Table 4.22 and it is seen that L-M method stays the same with the almost same MSE fit value. Quasi-Newton methods could preserve their performance as well and follows the L-M method. Consequently, Newton’s method and Nelder-Mead method fail to converge in the presence of poor initial values. They could not handle it and resulted in unacceptable fits. In terms of execution time and iteration number there is no significant difference between the algorithms as good and poor initials are considered. The order from the best to the worst is as L-M > BFGS > DFP > Newton > Nelder-Mead.

Simulated Data Case

4.2.2.1 Comparisons with respect to Error Distribution

In this section, the same path used in Section 4.2.1.1 will be pursued for the complex model this time. Other than the error distribution, all conditions are fixed and comparisons are conducted based on that. Firstly, sample size is fixed as large together with good initial values. Under these conditions, the results with 3 different error distributions are obtained separately.

Table 4.23: Large sample + normal error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-2.46	**1.81×10^{-3}**	-0.632	22.17	0.159
	β_2	-100.9	**-4.51×10^{-3}**	-2.288	-103.3	-7.393
	β_3	-71.84	**-2.41×10^{-3}**	-1.631	-62.04	-4.359
	β_4	-12.62	**-3.04×10^{-4}**	-0.316	-9.68	-0.766
	β_5	-9.29×10^{-2}	**-4.37×10^{-6}**	-1.53×10^{-3}	-9.51×10^{-2}	-4.53×10^{-3}
	β_6	-4.01×10^{-2}	**3.57×10^{-7}**	-1.54×10^{-3}	-6.29×10^{-3}	-1.33×10^{-3}
	β_7	-3.08×10^{-3}	**-7.21×10^{-7}**	-4.51×10^{-4}	-1.42×10^{-2}	-1.99×10^{-3}
MSE	β_1	80.06	**1.83×10^{-2}**	26.18	814.9	0.896
	β_2	10199.1	**0.023**	2186.4	13879.8	1426.17
	β_3	5174.8	**0.129**	1438.8	4183.1	505.945
	β_4	160.2	**4.34×10^{-3}**	58.28	105.4	15.645
	β_5	8.71×10^{-3}	**2.21×10^{-7}**	1.45×10^{-3}	0.011	5.64×10^{-4}
	β_6	1.63×10^{-3}	**5.89×10^{-8}**	4.45×10^{-4}	7.2×10^{-4}	5.22×10^{-5}
	β_7	4.73×10^{-5}	**2.47×10^{-9}**	6.19×10^{-5}	6.64×10^{-4}	1.01×10^{-4}
Time		0.0400	**0.004**	0.1752	0.1083	0.1255
Iterations		124.1	**9.04**	41.19	501.3	42.33
MSE (fit)		203.3	**0.97**	37.73	519.6	17.11

Table 4.24: Large sample + GL (b=0.5) error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-4.104	**-1.39**	-2.01	21.69	-1.335
	β_2	-100.9	**-13.5**	-3.31	-102.9	-4.047
	β_3	-71.50	**-0.554**	-1.91	-61.89	-1.661
	β_4	-12.53	**-0.069**	-0.338	-9.65	-0.213
	β_5	-0.091	**3.93×10^{-6}**	-1.04×10^{-3}	-0.094	-1.27×10^{-3}
	β_6	-0.039	**-1.44×10^{-6}**	-1.34×10^{-3}	-6.11×10^{-3}	4.55×10^{-5}
	β_7	-2.93×10^{-3}	**-1.59×10^{-6}**	-5.14×10^{-4}	-0.013	-1.06×10^{-3}
MSE	β_1	105.1	**2.05**	19.69	795.9	1.957
	β_2	10216.8	**3.29**	2097.1	13582.8	792.941
	β_3	5126.7	**1.16**	1409.5	4150.9	352.458
	β_4	157.9	**0.034**	59.29	105.1	11.765
	β_5	8.52×10^{-3}	**1.45×10^{-6}**	1.29×10^{-3}	0.011	4.40×10^{-4}
	β_6	1.61×10^{-3}	**3.89×10^{-7}**	4.53×10^{-4}	7.25×10^{-4}	9.54×10^{-5}
	β_7	4.52×10^{-5}	**1.64×10^{-8}**	-5.50×10^{-5}	6.45×10^{-4}	5.08×10^{-5}
Time		0.0096	**0.0025**	0.0308	0.0259	0.1059
Iterations		123.5	**8.94**	29.89	501.2	43.89
MSE (fit)		206.2	**6.41**	41.21	545.9	20.011

Table 4.25: Large sample + GL (b=2) error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-1.37	**1.01**	0.34	22.41	1.191
	β_2	-100.9	**0.963**	-1.28	-102.35	-8.856
	β_3	-71.99	**0.394**	-1.14	-61.71	-5.413
	β_4	-12.65	**0.049**	-0.24	-9.71	-0.959
	β_5	-0.093	**-8.88×10^{-6}**	-1.41×10^{-3}	-0.095	-6.18×10^{-3}
	β_6	-0.041	**4.73×10^{-7}**	-1.49×10^{-3}	-7.61×10^{-3}	-1.86×10^{-3}
	β_7	-3.28×10^{-3}	**6.26×10^{-7}**	-4.72×10^{-4}	-0.014	-2.51×10^{-3}
MSE	β_1	71.83	**1.050**	20.17	822.1	2.177
	β_2	10212.9	**1.432**	2239.5	13234.1	1959.151
	β_3	5200.7	**0.448**	1429.6	4094.7	704.857
	β_4	161.48	**0.012**	58.03	104.6	21.795
	β_5	8.84×10^{-3}	**5.03×10^{-7}**	1.41×10^{-3}	0.011	8.03×10^{-4}
	β_6	1.65×10^{-3}	**1.33×10^{-7}**	4.71×10^{-4}	6.58×10^{-4}	8.53×10^{-5}
	β_7	5.21×10^{-5}	**5.68×10^{-9}**	5.49×10^{-5}	6.45×10^{-4}	1.26×10^{-4}
Time		0.0301	**0.0053**	0.1231	0.1100	0.1300
Iterations		124.2	**9.01**	41.09	501.4	42.20
MSE (fit)		213.4	**2.23**	29.71	519.7	15.29

The corresponding results are given in Table 4.23, Table 4.24 and Table 4.25. Overall, L-M algorithm seems like the best one among them in estimating the parameters. Moreover, it can be said that the other algorithms do not work that good compared to the results of them with the simple model. Increase in the number of parameters and model complexity affects the performance of the algorithms negatively. However, DFP method and BFGS method still proposes plausible fits to the model even though they are not very close to the ones proposed by L-M method. On the other hand, Newton-Raphson method and Nelder-Mead method could not be successful in estimation for these conditions. Their MSE fit values are so high that the fits are unacceptable.

The results can be examined method by method with respect to performance, too. To start with, L-M method provides the best fit for each distribution. Non-normality of errors affects the accuracy of the fit very slightly, but still the fits are good enough. The method works best with normal errors and worst with the left skewed generalized logistic errors.

Moving on with DFP method, it is clear that the method offers the second best fit under both conditions. When the corresponding MSE fit values are checked, it is clearly seen that the method is very consistent. The biases and MSEs for the estimates seem plausible, too. Furthermore, the method works best with the right skewed generalized logistic errors.

In the third place, there is BFGS algorithm which comes from the same family of methods with DFP method. The same conclusion applies to BFGS method as well. As DFP method, it also works best with the right skewed generalized logistic errors which means that quasi-Newton methods are not affected from non-normality of the error term seriously for this problem. This may be a good feature when working with real life problems.

Newton's method is successful with the simple model under exact same conditions. However, complexity of the model struggles the algorithm to converge to optimal solution. For the complex model, Newton is clearly not a good choice. On the other hand, the method is not affected by the non-normality significantly, but this information is useless since it fails to solve the problem.

Finally, Nelder-Mead method is the worst one for this model and its suggested fits are extremely far from being the optimal fit. Hence, there is no need to make further comments on its results.

As in the simple model case, the conclusion with the poor initial values is not different from the one with good initial values. The same comments apply to that case as well. That is why the results of them are not presented in this section.

4.2.2.2 Comparisons with respect to Sample Size

In order to conduct the comparisons according to sample size, it is assumed that the errors are normal and initial values are good as in the simple case. The results are given in Table 4.23 and Table 4.26.

This scenario is considered as the best case scenario for this study since both errors and initial values are well-conditioned. According to the tabulated results obtained through simulation study, L-M method is the best one by far in both small and large sample sizes. It provides the estimates with least bias and MSE values along with the smallest MSE value for overall fit.

Except DFP, other methods satisfy statistical consistency. DFP competes with L-M for when sample size is small. However, since L-M manages it with smaller number of iterations within shorter amount of time, it can be concluded that it is the most successful one for this comparison case. Specifically, Nelder-Mead and Newton's methods fail to converge to optimal solution and could not propose reasonable fits.

Table 4.26: Small sample + normal error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	$\boldsymbol{\beta_1}$	-5.97	**6.81×10^{-4}**	-0.93	23.31	0.083
	$\boldsymbol{\beta_2}$	-101.5	**-0.081**	-5.85	-97.91	-2.741
	$\boldsymbol{\beta_3}$	-69.57	**-0.065**	-4.41	-61.97	-2.392
	$\boldsymbol{\beta_4}$	12.52	**-0.013**	-0.91	-9.98	-0.539
	$\boldsymbol{\beta_5}$	-0.083	**-7.16×10^{-5}**	-4.26×10^{-3}	-0.09	-2.16×10^{-3}
	$\boldsymbol{\beta_6}$	-0.038	**-4.25×10^{-5}**	-3.31×10^{-3}	-7.74×10^{-3}	-1.36×10^{-3}
	$\boldsymbol{\beta_7}$	-9.42×10^{-3}	**-8.26×10^{-6}**	-9.49×10^{-4}	-9.42×10^{-3}	-4.02×10^{-4}
MSE	$\boldsymbol{\beta_1}$	421.1	**0.217**	47.18	1006.2	0.204
	$\boldsymbol{\beta_2}$	10612.3	**26.45**	5119.5	11548.9	274.563
	$\boldsymbol{\beta_3}$	5053.3	**18.59**	2785.6	4072.1	201.735
	$\boldsymbol{\beta_4}$	174.9	**0.867**	109.6	107.1	10.23
	$\boldsymbol{\beta_5}$	7.29×10^{-3}	**1.75×10^{-5}**	3.14×10^{-3}	0.013	1.65×10^{-4}
	$\boldsymbol{\beta_6}$	1.91×10^{-3}	**6.67×10^{-6}**	8.57×10^{-4}	8.21×10^{-4}	6.91×10^{-5}
	$\boldsymbol{\beta_7}$	2.76×10^{-4}	**6.02×10^{-7}**	1.64×10^{-4}	8.97×10^{-4}	6.93×10^{-6}
Time		0.0042	**0.0014**	0.0197	0.0189	0.0525
Iterations		116.4	**10.45**	38.01	497.4	44.91
MSE (fit)		232.6	**0.752**	50.71	716.3	0.756

As clearly seen from the results, unlike the results of simple model case, other than L-M and DFP methods, decrease in sample size affects the performance of the algorithms in a negative way.

For the next comparison, the error distribution is selected as GL with the shape parameter b=2 and the initial values are close to the true solution, i.e., good. Examining Table 4.25 and Table 4.27, it can be clearly seen that L-M method is the best one again. It is the fastest with least number of iterations and least biased estimates among all algorithms.

DFP method provides the second best fit in both tables, but smaller sample size improves its performance in estimation. On contrary, BFGS method gets worse unlike DFP method when the sample size is converted from large to small. As a result, it is not a good choice for small samples under such conditions. Finally, Nelder-Mead and Newton methods fail for this case.

Their fits are nowhere near to a reasonable solution. Furthermore, if the results are interpreted with respect to consistency, only BFGS and Nelder-Mead methods produce better estimates when the sample size increases. This implies that they produce consistent estimators. On the contrary, this conclusion does not apply to the rest of the algorithms.

Table 4.27: Small sample + GL (b=2) error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	$\boldsymbol{\beta_1}$	-4.34	**0.993**	0.067	23.24	0.965
	$\boldsymbol{\beta_2}$	-101.4	**0.744**	-5.774	-98.32	-2.458
	$\boldsymbol{\beta_3}$	-69.99	**0.220**	-4.792	-62.21	-1.914
	$\boldsymbol{\beta_4}$	-12.61	**0.012**	-1.041	-10.01	-0.393
	$\boldsymbol{\beta_5}$	-0.084	**-1.74x10^{-4}**	-4.93x10^{-3}	-0.101	-2.48x10^{-3}
	$\boldsymbol{\beta_6}$	-0.038	**-1.06x10^{-4}**	-3.77x10^{-3}	-9.19x10^{-3}	-1.13x10^{-3}
	$\boldsymbol{\beta_7}$	-9.28x10^{-3}	**-3.28x10^{-5}**	-1.03x10^{-3}	-9.51x10^{-3}	-6.61x10^{-4}
MSE	$\boldsymbol{\beta_1}$	374.5	**1.71**	46.36	996.7	1.436
	$\boldsymbol{\beta_2}$	10552.1	**118.8**	3540.8	11554.4	376.451
	$\boldsymbol{\beta_3}$	5107.1	**91.80**	2229.8	4103.3	190.925
	$\boldsymbol{\beta_4}$	177.9	**4.92**	97.03	108.5	7.468
	$\boldsymbol{\beta_5}$	7.43x10^{-3}	**7.52x10^{-5}**	2.28x10^{-3}	0.013	2.08x10^{-4}
	$\boldsymbol{\beta_6}$	1.92x10^{-3}	**3.31x10^{-5}**	7.83x10^{-4}	8.22x10^{-4}	4.62x10^{-5}
	$\boldsymbol{\beta_7}$	2.68x10^{-4}	**3.09x10^{-6}**	1.16x10^{-4}	8.76x10^{-4}	1.42x10^{-5}
Time		0.0041	**0.0019**	0.0198	0.0189	0.0486
Iterations		117.0	**10.47**	37.95	497.5	43.29
MSE (fit)		209.8	**1.806**	52.12	713.9	3.092

Last but not least, final comparison is applied on the simulated datasets whose error distribution is GL with shape parameter b=0.5 and the iteration process is started with the set of poor initial values. The related outputs are given in Tables 4.28 and 4.29, respectively. In this case, only L-M method is successful in estimating the model fit because other algorithms have very high MSE fit values. Although DFP works well with small sample size and good initial values, it fails in the presence of poor initial values. They all produce highly biased estimates and they are not even acceptable. This situation can be interpreted as convergence failure because they are not even close to the certified values.

Table 4.28: Large sample + GL (b=0.5) error + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	3.591	**-1.387**	-8.087	-136.06	0.981
	β_2	-243.9	**-1.344**	-26.70	-643.97	-88.991
	β_3	-152.9	**-0.552**	-20.27	-196.13	-58.939
	β_4	-25.71	**-0.069**	-4.05	-24.55	-10.486
	β_5	-0.186	**-1.54x10^{-6}**	-0.022	-0.353	-0.066
	β_6	-0.059	**-7.43x10^{-7}**	-0.015	0.034	-0.025
	β_7	-0.037	**-1.30x10^{-6}**	-1.64x10^{-3}	-0.182	-0.014
MSE	β_1	380.3	**2.044**	2592.7	47481.4	26.705
	β_2	11858.6	**3.291**	25887.9	497516.5	44951.05
	β_3	44723.1	**1.162**	16904.1	68829.1	20321.15
	β_4	1388.7	**0.034**	726.5	5307.1	655.56
	β_5	0.067	**1.47x10^{-6}**	0.017	0.244	0.026
	β_6	9.94x10^{-3}	**3.91x10^{-7}**	7.04x10^{-3}	0.126	3.84x10^{-3}
	β_7	3.12x10^{-3}	**1.61x10^{-8}**	5.62x10^{-4}	0.046	1.12x10^{-3}
Time		0.0175	**0.0026**	0.0393	0.0311	0.0901
Iterations		134.5	**11.18**	41.33	501.4	48.11
MSE (fit)		668.83	**6.391**	1504.2	19566.5	124.39

Table 4.29: Small sample + GL (b=0.5) error + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-4.607	**-1.443**	-10.953	-149.678	-5.772
	β_2	-977.4	**0.902**	-0.369	-629.691	-57.73
	β_3	-832.9	**0.574**	-2.155	-219.169	-32.66
	β_4	-172.7	**0.099**	-0.383	-31.935	-5.091
	β_5	-0.758	**1.74x10^{-3}**	-5.28x10^{-3}	-0.353	-0.043
	β_6	-0.490	**3.51x10^{-5}**	-9.67x10^{-3}	-0.023	-0.013
	β_7	-0.145	**5.16x10^{-4}**	7.11x10^{-3}	-0.175	-8.97x10^{-3}
MSE	β_1	66484.7	**4.667**	3555.4	51691.2	2245.18
	β_2	5.35x10^{9}	**3941.6**	41384.9	4.60x10^{5}	40719.87
	β_3	4.59x10^{9}	**1666.5**	21563.1	75487.1	12156.55
	β_4	2.14x10^{8}	**54.37**	832.0	4541.1	366.52
	β_5	3195.1	**2.35x10^{-3}**	0.025	0.227	0.025
	β_6	1758.1	**4.10x10^{-4}**	8.34x10^{-3}	0.121	2.97x10^{-3}
	β_7	126.3	**1.17x10^{-4}**	1.45x10^{-3}	0.062	1.92x10^{-3}
Time		0.0039	**0.0018**	0.0239	0.0235	0.0989
Iterations		131.5	**12.71**	45.52	499.1	47.87
MSE (fit)		1494.5	**7.557**	1316.4	18193.6	1611.6

To conclude, we can say that L-M algorithm is the only one to survive in each case. In addition to that, DFP method can be counted as a choice when the sample size is small and the initial values are selected properly. Other than that, the others are not plausible to be used in such situations.

4.2.2.3 Comparisons with respect to Initial Values

As in the case of simple model, the algorithms are compared with respect to goodness of initial values as well. The first case is when the errors are normally distributed and the sample size is large. As mentioned earlier, this is considered as the ideal case. The results for the cases that is solved by assigning good and bad initial values are presented in Tables 4.23 and 4.30.

Table 4.30: Large sample + normal error + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	4.863	**1.81×10^{-3}**	-7.856	-138.09	2.135
	β_2	-243.8	**-4.51×10^{-3}**	-25.89	-644.4	-66.767
	β_3	-153.6	**-2.41×10^{-3}**	-19.62	-197.14	-39.083
	β_4	-25.93	**-3.04×10^{-4}**	-3.891	-24.72	-6.158
	β_5	-0.188	**4.37×10^{-6}**	-0.023	-0.353	-0.051
	β_6	-0.061	**3.57×10^{-7}**	-0.016	-0.030	-0.013
	β_7	-0.037	**7.21×10^{-7}**	-1.34×10^{-3}	-0.181	-0.011
MSE	β_1	352.9	**0.018**	2870.8	48362.1	49.915
	β_2	111881	**0.222**	25740.5	499200	39142.8
	β_3	45031	**0.129**	16078.5	67691.1	12180.2
	β_4	1403.3	**4.34×10^{-3}**	669.1	4956.6	373.88
	β_5	0.068	**2.21×10^{-7}**	0.017	0.247	0.024
	β_6	8.96×10^{-3}	**5.89×10^{-8}**	6.98×10^{-3}	0.121	2.47×10^{-3}
	β_7	3.13×10^{-3}	**2.47×10^{-9}**	5.28×10^{-4}	0.045	9.20×10^{-4}
Time		0.0323	**0.0052**	0.1260	0.1145	0.1218
Iterations		134.5	**11.21**	41.39	501.5	45.83
MSE (fit)		690.25	**0.972**	1654.3	19820.3	117.837

When results in Table 4.23 is analyzed, it can be seen that L-M achieved pretty good convergence with very few number of iterations. The biases and MSE of the estimates are negligibly small. Levenberg-Marquarth, DFP and BFGS methods provide fits that may be considered as reasonable. The rest of the algorithms fail to converge.

On the other hand, when the case gets more complex with the assignment of poor initial values at the beginning of the process, the results in Table 4.30 are observed. As it is crystal clear that only L-M method could achieve to stay still. It only does the job with more number of iterations, which is not very significant as long as the bias and MSEs are good. The others fail drastically and produce highly biased estimates with high MSE values.

Second comparison is done under the case in which the data is simulated with GL errors with shape parameter b=2 and large sample size. The corresponding results are presented in the Tables 4.25 and 4.31.

Table 4.31: Large sample + GL (b=2) error + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	6.376	**1.004**	-6.196	-133.448	3.837
	β_2	-244.86	**0.963**	-23.362	-644.72	-93.267
	β_3	-153.07	**0.394**	-17.866	-195.694	-53.037
	β_4	-25.723	**0.049**	-3.570	-25.282	-8.102
	β_5	-0.189	**-8.88×10^{-6}**	-0.021	-0.356	-0.073
	β_6	-0.059	**4.73×10^{-7}**	-0.014	0.036	-0.016
	β_7	-0.037	**6.26×10^{-7}**	-1.09×10^{-3}	-0.183	-0.015
MSE	β_1	323.6	**1.050**	2763.1	47444.1	86.308
	β_2	112848.4	**1.432**	23001.7	4.96×10^{5}	57821.18
	β_3	44816.1	**0.448**	14367.6	67993.1	15658.45
	β_4	1399.1	**0.012**	605.15	5018.6	481.58
	β_5	0.068	**5.03×10^{-7}**	0.016	0.246	0.036
	β_6	9.04×10^{-3}	**1.33×10^{-7}**	6.27×10^{-3}	0.125	3.71×10^{-3}
	β_7	3.16×10^{-3}	**5.68×10^{-9}**	4.75×10^{-4}	0.046	1.31×10^{-3}
Time		0.0358	**0.0055**	0.1351	0.1161	0.1023
Iterations		134.5	**11.19**	41.25	501.51	48.33
MSE (fit)		652.53	**2.225**	1579.74	19761.01	164.93

Looking at the tabulated results for both good and poor initial value cases, it is seen that nothing is different than the previous case. Only the results get worse with the involvement of non-normal errors. Hence, all comments made on previous comparison case is valid for this case, too. L-M method is the only one to overcome the obstacle of poorly selected starting values.

Finally, the results of the simulated data for which error distribution is GL (b=0.5) and sample size is small are discussed. This case is considered as the worst case among all the simulation scenarios used in this study. Tables 4.29 and 4.32 illustrate the outputs under the same distributional and sample size assumption but for poor and good initial values, respectively.

Table 4.32: Small sample + GL (b=0.5) error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-7.56	**-1.38**	-2.21	23.08	-1.278
	β_2	-101.6	**-1.68**	-7.31	-97.71	-7.810
	β_3	-69.21	**-0.811**	-5.25	-62.05	-5.169
	β_4	-12.41	**-0.121**	-1.07	-9.98	-0.988
	β_5	-0.082	**-2.62×10^{-4}**	-4.46×10^{-3}	-0.101	-4.64×10^{-3}
	β_6	-0.038	**-1.44×10^{-4}**	-3.43×10^{-3}	-7.29×10^{-3}	-2.02×10^{-3}
	β_7	-9.21×10^{-3}	**-5.02×10^{-5}**	-9.42×10^{-4}	-9.03×10^{-3}	-1.22×10^{-3}
MSE	β_1	446.4	**3.43**	57.12	1036.5	4.187
	β_2	10751.8	**267.2**	3384.4	11325.9	823.951
	β_3	5098.1	**201.9**	2151.6	4077.9	389.023
	β_4	177.1	**10.73**	93.17	106.8	15.081
	β_5	7.18×10^{-3}	**1.69×10^{-4}**	2.14×10^{-3}	0.013	3.88×10^{-4}
	β_6	1.91×10^{-3}	**7.41×10^{-5}**	7.65×10^{-4}	8.28×10^{-4}	8.77×10^{-7}
	β_7	2.65×10^{-4}	**6.74×10^{-6}**	1.21×10^{-4}	8.61×10^{-4}	4.09×10^{-5}
Time		0.0043	**0.0012**	0.0193	0.0187	0.0508
Iterations		116.2	**10.48**	37.72	496.94	40.26
MSE (fit)		218.7	**4.92**	62.82	766.6	13.235

The results confirm that this case is the hardest one to deal with because even L-M is affected. In the previous cases, L-M method did not offer a different result when poor initials are assigned to the process.

However, this time its MSE for fit value slightly increased. Nevertheless, its fit is still plausible compared to the others. Moreover, DFP method seems to offer the second best fit in Table 4.32, but when poor initials are assigned, it fails badly just like the rest of the algorithms.

4.2.2.4 Comparisons with respect to Robustness

In this final section of comparisons, the robustness of algorithms to outliers, contamination and inliers in error term will be tested as it was done in Section 4.2.1.4 for the simple model.

Comparisons with respect to Robustness to Outliers in Errors

The procedure is exactly the same with the one for the simple model, i.e., 90% of the errors are generated from N(0,1) and 10% from N(0,4). To test the performance of algorithms with the presence of outliers in the error distribution, the ideal conditioned case is examined. The tabulated values are given in Tables 4.33 and 4.34.

Table 4.33: Large sample + error with outliers + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-2.653	**5.51x10^{-3}**	-0.398	22.538	5.81x10^{-2}
	β_2	-100.834	**-3.48x10^{-2}**	3.467	-102.814	-4.108
	β_3	-71.616	**-2.01x10^{-2}**	2.496	-60.578	-2.294
	β_4	-12.569	**-2.95x10^{-3}**	0.462	-9.337	-0.385
	β_5	-9.21x10^{-2}	**-3.74x10^{-5}**	2.72x10^{-3}	-9.36x10^{-2}	-2.48x10^{-3}
	β_6	-4.01x10^{-2}	**5.19x10^{-6}**	7.11x10^{-4}	-4.89x10^{-3}	-6.15x10^{-4}
	β_7	-3.47x10^{-3}	**3.01x10^{-6}**	4.11x10^{-4}	-1.48x10^{-2}	-1.12x10^{-3}
MSE	β_1	94.933	**2.35x10^{-2}**	3.091	757.01	0.306
	β_2	10174.2	**0.279**	1121.89	13124.65	594.91
	β_3	5134.51	**0.157**	471.81	4033.59	178.33
	β_4	158.292	**5.28x10^{-3}**	15.968	108.85	5.052
	β_5	8.55x10^{-3}	**2.44x10^{-7}**	3.91x10^{-4}	1.04x10^{-2}	2.08x10^{-4}
	β_6	1.62x10^{-3}	**6.82x10^{-8}**	1.01x10^{-4}	7.73x10^{-4}	1.42x10^{-5}
	β_7	4.19x10^{-5}	**3.16x10^{-9}**	5.87x10^{-5}	6.22x10^{-4}	5.18x10^{-5}
Time		0.0143	**0.0031**	0.0361	0.0305	0.0508
Iterations		124.0	**8.98**	40.92	501.55	40.57
MSE (fit)		191.31	**1.28**	17.02	502.86	11.79

If Table 4.33 is compared with Table 4.23 which covers the results of the ideal case but normally distributed errors having no outliers, it can be seen that the MSE for fit values and iteration numbers are more or less the same for each algorithm. According to the results in the table, L-M algorithm provide the best fit again. The method is fast to converge and needs only 8.98 iterations in average. Following it, quasi-Newton methods provide plausible fits looking at their MSE for overall fit values. The others are not good, especially Nelder-Mead. It is not unexpected because Nelder-Mead fails from the beginning of the complex model. The method could not deal with the complexity of the model.

The tabulated results for poor initial values are given in Table 4.34.

Table 4.34: Large sample + error with outliers + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	3.381	**3.17x10^{-2}**	-1.951	-164.87	2.456
	β_2	-236.87	**4.24x10^{-2}**	-3.611	-663.62	-94.74
	β_3	-149.62	**1.81x10^{-2}**	-2.991	-207.64	-62.06
	β_4	-25.31	**2.48x10^{-3}**	-0.649	-28.73	-10.99
	β_5	-0.185	**1.11x10^{-5}**	-3.23x10^{-3}	-0368	-7.01x10^{-2}
	β_6	-5.99x10^{-2}	**2.55x10^{-5}**	-4.04x10^{-3}	4.31x10^{-3}	-2.55x10^{-2}
	β_7	-3.49x10^{-2}	**6.45x10^{-6}**	7.93x10^{-5}	-0.177	-1.64x10^{-2}
MSE	β_1	507.78	**2.55x10^{-2}**	141.51	58420.4	29.09
	β_2	109006.8	**0.296**	4308.2	552614.1	45149.74
	β_3	43929.8	**0.173**	2506.3	78792.0	20320.6
	β_4	1362.6	**5.89x10^{-3}**	103.28	8798.7	654.31
	β_5	6.96x10^{-2}	**3.22x10^{-7}**	2.72x10^{-3}	0.300	2.59x10^{-2}
	β_6	8.13x10^{-3}	**7.96x10^{-8}**	1.59x10^{-3}	0.137	3.87x10^{-3}
	β_7	3.12x10^{-3}	**3.59x10^{-9}**	7.73x10^{-5}	4.24x10^{-2}	1.19x10^{-3}
Time		0.0223	**0.0035**	0.0367	0.0322	0.0570
Iterations		134.46	**11.35**	42.59	501.21	41.17
MSE (fit)		928.4	**1.27**	307.84	22190.8	147.34

When the results in Table 4.34 are checked on its own, it can be seen that only L-M method produces a good fit. The others are not plausible at all. There is no doubt about the conclusion that L-M method is the only option for such cases.

Moreover, the results in the absence of outliers is given in Table 4.30. If the performances of algorithms are compared, it is obvious that all methods get worse. Actually, they are not good when the outliers are absent either. Hence, it can be concluded that only L-M could handle the outliers and poor initial values together.

Comparisons with respect to Robustness to Contamination in Errors

In this part, the aim is to observe the change in the performance of the algorithms due to contamination in errors.

Table 4.35: Large sample + contaminated error + good initial values (Thurber)

		Newton	**L-M**	**BFGS**	**Nelder-Mead**	**DFP**
Bias	β_1	-2.189	**5.15x10^{-2}**	-0.621	22.87	0.102
	β_2	-101.1	**5.90x10^{-2}**	-0.918	-104.4	-1.194
	β_3	-71.77	**2.58x10^{-2}**	-0.438	-62.71	-0.345
	β_4	-12.57	**3.34x10^{-3}**	-7.19x10^{-2}	-9.756	-4.40x10^{-3}
	β_5	-9.29x10^{-2}	**1.02x10^{-5}**	-2.64x10^{-4}	-9.75x10^{-2}	-9.50x10^{-4}
	β_6	-3.97x10^{-2}	**4.03x10^{-6}**	-7.97x10^{-4}	-6.80x10^{-3}	2.01x10^{-4}
	β_7	-3.31x10^{-3}	**5.22x10^{-7}**	-3.17x10^{-4}	-1.40x10^{-2}	-2.62x10^{-4}
MSE	β_1	57.86	**2.01x10^{-2}**	8.182	866.2	1.014
	β_2	10237.1	**0.212**	217.01	13977.4	920.6
	β_3	5164.8	**0.121**	119.1	4223.7	597.7
	β_4	159.36	**4.05x10^{-3}**	3.869	104.9	21.51
	β_5	8.70x10-3	**2.03x10^{-7}**	1.94x10^{-4}	1.18x10^{-2}	9.28x10^{-4}
	β_6	1.61x10^{-3}	**5.42x10^{-8}**	3.77x10^{-5}	7.01x10^{-4}	2.10x10^{-4}
	β_7	5.07x10^{-5}	**2.45x10^{-9}**	7.63x10^{-6}	6.88x10^{-4}	7.93x10^{-6}
Time		0.0143	**0.0027**	0.0389	0.0314	0.0344
Iterations		124.1	**9.09**	41.1	501.2	42.07
MSE (fit)		190.41	**0.905**	19.89	548.47	28.64

Table 4.36: Small sample + contaminated error + good initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-5.492	**8.52×10^{-2}**	-0.677	22.05	-4.23×10^{-2}
	β_2	-102.3	**3.03×10^{-2}**	-6.181	-102.63	-3.441
	β_3	-69.82	**5.56×10^{-3}**	-4.831	-62.45	-2.439
	β_4	-12.54	**1.79×10^{-3}**	-0.992	-9.741	-0.501
	β_5	-8.37×10^{-2}	**-7.24×10^{-5}**	-4.66×10^{-3}	-0.103	-2.12×10^{-3}
	β_6	-3.78×10^{-2}	**5.05×10^{-6}**	-3.36×10^{-3}	-7.46×10^{-3}	-1.14×10^{-3}
	β_7	-9.79×10^{-3}	**1.86×10^{-5}**	-8.89×10^{-4}	-1.05×10^{-2}	-8.27×10^{-4}
MSE	β_1	377.1	**0.208**	47.23	892.8	2.637
	β_2	10561.3	**20.87**	2404.6	13501.2	392.6
	β_3	4994.5	**13.75**	1750.8	4309.4	195.6
	β_4	171.6	**0.576**	73.45	119.5	8.402
	β_5	7.23×10^{-3}	**1.68×10^{-5}**	1.56×10^{-3}	1.38×10^{-2}	1.40×10^{-4}
	β_6	1.83×10^{-3}	**4.53×10^{-6}**	6.63×10^{-4}	9.38×10^{-4}	4.73×10^{-5}
	β_7	2.44×10^{-4}	**3.57×10^{-7}**	9.37×10^{-5}	1.06×10^{-3}	3.15×10^{-5}
Time		0.0038	**0.0013**	0.0241	0.0256	0.0128
Iterations		116.5	**10.46**	37.82	498.6	44.86
MSE (fit)		228.87	**0.777**	60.18	707.8	2.924

Table 4.37: Large sample + contaminated error + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	6.201	**5.40x10^{-2}**	-4.861	-134.1	3.101
	β_2	-241.3	**4.25x10^{-2}**	-24.68	-649.4	-97.12
	β_3	-150.3	**1.65x10^{-2}**	-20.26	-197.2	-56.47
	β_4	-25.23	**2.12x10^{-3}**	-4.186	-25.12	-8.868
	β_5	-0.184	**-3.69x10^{-6}**	-2.11x10^{-2}	-0.361	-7.14x10^{-2}
	β_6	-5.74x10^{-2}	**5.59x10^{-6}**	-1.49x10^{-2}	3.62x10^{-2}	-1.89x10^{-2}
	β_7	-3.83x10^{-2}	**8.69x10^{-7}**	-1.86x10^{-3}	-0.184	-1.66x10^{-2}
MSE	β_1	147.3	**1.99x10^{-2}**	1788.3	46678.5	104.7
	β_2	111260.1	**0.210**	24210.9	508177.7	58643.9
	β_3	43948.6	**0.121**	16872.5	65571.1	16581.5
	β_4	1362.3	**4.07x10^{-3}**	737.5	4922.8	472.8
	β_5	6.52x10^{-2}	**2.11x10^{-7}**	1.59x10^{-2}	0.256	3.03x10^{-2}
	β_6	8.73x10^{-3}	**5.16x10^{-8}**	6.79x10^{-3}	0.121	3.19x10^{-2}
	β_7	3.12x10^{-3}	**2.29x10^{-9}**	5.42x10^{-4}	4.58x10^{-2}	1.74x10^{-3}
Time		0.0214	**0.0033**	0.0441	0.0354	0.0519
Iterations		134.6	**11.17**	41.61	501.5	47.32
MSE (fit)		496.9	**0.906**	1193.1	19707	230.1

Table 4.38: Small sample + contaminated error + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-5.508	**-8.84×10^{-2}**	-12.23	157.6	-21.53
	β_2	-233.2	**5.392**	1.437	-608.7	-77.56
	β_3	146.6	**2.668**	-0.196	214.7	-47.99
	β_4	-24.97	**0.362**	0.138	-32.92	-8.411
	β_5	-0.182	**4.27×10^{-3}**	-2.36×10^{-3}	-0.329	-6.23×10^{-2}
	β_6	-6.59×10^{-2}	**1.39×10^{-4}**	-1.13×10^{-2}	-3.45×10^{-2}	-2.72×10^{-2}
	β_7	3.22×10^{-2}	**1.08×10^{-3}**	6.83×10^{-3}	0.173	-1.08×10^{-2}
MSE	β_1	4216.1	**2.01**	4718.1	54424.8	9551.1
	β_2	153851.9	**4071.3**	45717.1	420743.8	36311.7
	β_3	64246.2	**1211.7**	24219.9	68384.1	14553.1
	β_4	2184.7	**30.71**	1152.5	3253.4	486.9
	β_5	9.32×10^{-2}	**2.62×10^{-3}**	2.51×10^{-2}	0.201	2.41×10^{-2}
	β_6	1.76×10^{-2}	**9.55×10^{-5}**	9.43×10^{-3}	0.107	4.97×10^{-3}
	β_7	4.32×10^{-3}	**1.49×10^{-4}**	2.09×10^{-3}	5.91×10^{-2}	6.86×10^{-4}
Time		0.0058	**0.0016**	0.0261	0.0250	0.0139
Iterations		131.8	**12.81**	43.22	499.6	44.5
MSE (fit)		1504.3	**6.232**	1470.5	18644	7435.3

Firstly, when the results of the cases with and without contamination in error term are compared, it can be concluded that the results are pretty much consistent. Undoubtedly, the fits that are obtained from the datasets without contamination are better. Yet, L-M algorithm is the only one which is not affected from contamination, significantly. The method only worsens with the case of poor initial values and small sample size. However, there is no dramatic change and the fit is still better than the ones provided by the rest of the algorithms.

The other algorithms could not deal with the contamination and fails to produce a plausible fit. Among them, Nelder-Mead algorithm is the worst one overall. It is not unexpected because Nelder-Mead algorithm did not do a great job with the complex model in all cases anyway. To conclude, it can be stated that only L-M method becomes successful in the presence of contamination in errors.

Comparisons with respect to Robustness to Inliers in Errors

The final comparison condition for the complex model is the presence of inliers in the error term. Again, only the results with poor initial values will be given since there is no significant difference between them.

Table 4.39: Large sample + error with inliers + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	5.024	**-1.43x10^{-4}**	-7.109	-120.27	3.461
	β_2	-239.02	**-6.56x10^{-3}**	-26.96	-665.39	-116.19
	β_3	-149.57	**-4.40x10^{-3}**	-20.91	-191.27	-68.117
	β_4	-25.13	**-7.54x10^{-4}**	-4.171	-23.575	-10.723
	β_5	-0.184	**-2.70x10^{-6}**	-2.48x10^{-2}	-0.381	-8.90x10^{-2}
	β_6	-5.83x10^{-2}	**-3.46x10^{-6}**	-1.60x10^{-2}	6.48x10^{-2}	-2.29x10^{-2}
	β_7	-3.69x10^{-2}	**-1.59x10^{-6}**	-7.57x10^{-4}	-0.191	-1.83x10^{-2}
MSE	β_1	349.0	**1.17x10^{-2}**	2906.7	43299.6	82.79
	β_2	109991.0	**0.118**	32012.0	541064.5	64169.5
	β_3	43761.5	**6.69x10^{-2}**	20002.1	76367.5	18275.0
	β_4	1360.7	**2.23x10^{-3}**	836.4	6495.7	524.43
	β_5	6.61x10^{-2}	**1.12x10^{-7}**	2.16x10^{-2}	0.277	3.96x10^{-2}
	β_6	8.66x10^{-3}	**2.88x10^{-8}**	8.06x10^{-3}	0.141	3.54x10^{-3}
	β_7	3.10x10^{-3}	**1.33x10^{-9}**	6.89x10^{-4}	5.01x10^{-2}	1.46x10^{-3}
Time		0.0128	**0.003**	0.0366	0.0325	0.0508
Iterations		134.94	**11.21**	41.59	501.5	42.59
MSE (fit)		634.4	**0.541**	1882.9	18936.0	197.8

Table 4.40: Small sample + error with inliers + poor initial values (Thurber)

		Newton	L-M	BFGS	Nelder-Mead	DFP
Bias	β_1	-5.437	**-0.134**	-6.684	-150.48	-2.242
	β_2	-233.45	**3.750**	5.187	-642.43	-46.27
	β_3	-146.14	**2.132**	0.671	-226.79	-16.27
	β_4	-24.77	**0.348**	$-4.42\text{x}10^{-3}$	-32.46	-3.558
	β_5	-0.183	$\mathbf{2.92\text{x}10^{-3}}$	$-3.21\text{x}10^{-3}$	-0.374	$-3.21\text{x}10^{-2}$
	β_6	$-6.49\text{x}10^{-2}$	$\mathbf{5.29\text{x}10^{-4}}$	$-7.33\text{x}10^{-3}$	$-2.48\text{x}10^{-2}$	$3.58\text{x}10^{-3}$
	β_7	$-3.16\text{x}10^{-2}$	$\mathbf{8.11\text{x}10^{-4}}$	$7.16\text{x}10^{-3}$	-0.171	$-1.21\text{x}10^{-2}$
MSE	β_1	4076.3	**0.981**	3450.8	53528.5	1500.7
	β_2	159080.6	**7823.9**	41477.7	482584.2	37123.3
	β_3	67431.3	**2684.2**	20583.6	75529.1	8258.3
	β_4	2343.2	**71.91**	753.41	4210.8	246.15
	β_5	$9.74\text{x}10^{-2}$	$\mathbf{4.61\text{x}10^{-3}}$	$2.64\text{x}10^{-2}$	0.251	$1.72\text{x}10^{-2}$
	β_6	$1.90\text{x}10^{-2}$	$\mathbf{3.24\text{x}10^{-4}}$	$7.53\text{x}10^{-3}$	0.114	$2.10\text{x}10^{-2}$
	β_7	$4.61\text{x}10^{-3}$	$\mathbf{2.70\text{x}10^{-4}}$	$1.37\text{x}10^{-3}$	$6.01\text{x}10^{-2}$	$5.38\text{x}10^{-3}$
Time		0.0033	**0.0021**	0.0247	0.0230	0.0113
Iterations		131.87	**12.64**	43.58	498.39	44.71
MSE (fit)		1525.2	**4.291**	1098.1	18694.4	685.8

In Table 4.39, biases and MSEs are observed as very high for all algorithms except for L-M method. L-M method produces practically unbiased estimates with low variances. Looking at the MSE for overall fit values, it can be easily concluded that L-M method is the only one that suggests a plausible, actually nearly perfect fit. The others' MSE fit values are so high that makes them unacceptable. In addition to that, L-M method is the fastest one again with 0.0021 seconds, which is very fast.

On the other hand, Table 4.40 presents the results for the small sample case. In Table 4.40, biases and MSEs increase dramatically compared to the large sample case, again except for L-M method. Nevertheless, there also is a slight increase in the bias and MSE of L-M method. Despite that, L-M method is still the best one because bias and MSE values are still the smallest. Moreover, its MSE for overall fit value implies that the fit is very good, indeed. All comments for the former case apply to this one as well. As a results, L-M method is the only option for such case.

CHAPTER 5

CONCLUSION

In this thesis, iterative methods that is commonly used for solving the parameter estimation problems in nonlinear regression analysis are briefly explained and comparative study on them is conducted with respect to several criteria under several conditions. The aim is to see the performances of these numerical algorithms under such conditions and superiorities over each other. Comments on the methods are given with respect to their performance.

To be able to comment on their performances, Monte Carlo simulation study that covers all possible situations is conducted. As a result of that, it is concluded that L-M method is the most successful method among the others under almost all situations that we have considered. It produces the estimates for parameters not only with the least bias and MSE but also with least number of iterations and within shortest execution time. The method performs quite well with non-normal errors, poor initial values, complex models, outliers etc.

DFP method works quite good with the simple model under any condition. Under cases with good initial values, it provides the best fit together with L-M and Nelder-Mead methods. On the other hand, it gets worse as the conditions get more complex. For instance, the method fails when the model is complex with poor initial values.

Newton's method works well with the simple model, especially when the initial values are close enough to the optimal solution. On the other hand, it could not handle the complex model, properly. The performance of the method gets worse as the number of parameters in the model increases.

BFGS method comes from the same family of methods as DFP method belongs to. However, it could not be as successful as DFP method. In general, under all cases considered in this study, the performance of BFGS method is moderate.

Nelder-Mead algorithm which is slightly different from the others in the sense making no use of derivatives performs very well with the simple model, especially with good initial values. It also results in plausible fits with poor initial guesses. On the other hand, it becomes the worst one for the complex model. The reason is that as the number of parameters increases, simplex used in the procedure becomes more complex due to increase in the dimension. As a result, it is not recommended to be used in models with high number of parameters.

Finally, nonlinear conjugate gradient can be concluded as the worst one, overall. The method is so unsuccessful with the simple model that it is not included in the simulation study for complex model.

To make a general comment on the findings, it can be stated that some algorithms could easily converge to the global minimum while the others get stuck in the local minimum. The reason is that the algorithms used in this study are local search methods. Hence, there is a possibility to converge to local optima, which is not as desired. In the literature, there are global search methods such as grid search and genetic algorithm. They will be the subject of our further study.

The summary results of L-M method for each 24 simulation scenarios are presented in Table 4.41. As it is clear, the method produced quite good results on any case that is considered in this study. To conclude, L-M method is the most preferable algorithm according to our simulation study and real data application results due to its efficiency, precision, robustness and speed in nonlinear parameter estimation.

Table 4.41: Summary results for L-M method

Chwirut1		Thurber	
	MSE (fit)		**MSE (fit)**
1.scenario	0.99	**1.scenario**	0.97
2.scenario	0.88	**2.scenario**	0.75
3.scenario	6.54	**3.scenario**	6.41
4.scenario	5.86	**4.scenario**	4.92
5.scenario	2.28	**5.scenario**	2.20
6.scenario	2.06	**6.scenario**	1.80
7.scenario	0.99	**7.scenario**	0.97
8.scenario	0.87	**8.scenario**	4.45
9.scenario	6.58	**9.scenario**	6.39
10.scenario	5.77	**10.scenario**	7.55
11.scenario	2.27	**11.scenario**	2.22
12.scenario	2.08	**12.scenario**	5.82
13.scenario	1.31	**13.scenario**	1.28
14.scenario	1.19	**14.scenario**	1.06
15.scenario	6.05	**15.scenario**	1.27
16.scenario	38.57	**16.scenario**	0.97
17.scenario	0.92	**17.scenario**	0.90
18.scenario	0.87	**18.scenario**	0.77
19.scenario	0.91	**19.scenario**	0.90
20.scenario	1.14	**20.scenario**	6.23
21.scenario	0.55	**21.scenario**	0.53
22.scenario	0.49	**22.scenario**	0.40
23.scenario	1.54	**23.scenario**	0.54
24.scenario	5.68	**24.scenario**	4.29

REFERENCES

Akkaya, A.D. & Tiku, M.L. TEST (2008) 17: 282. https://doi.org/10.1007/s11749-006-0032-8

Bates, D.M., & Watts, D. G. (2007). Nonlinear regression analysis and its applications (2nd ed.). New York: Wiley.

Broyden, C. G. (1965). A Class of Methods for Solving Nonlinear Simultaneous Equations. Mathematics of Computation. American Mathematical Society. 19 (92): 577–593.

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms, Journal of the Institute of Mathematics and Its Applications, 6, 76–90.

Byrd, Richard H. (1996). Analysis of a Symmetric Rank-One Trust Region Method. SIAM Journal on Optimization 6(4).

Chwirut, D., NIST (1979). Ultrasonic Reference Block Study.

Curry, H. B. (1944). The method of steepest descent for nonlinear minimization problems, Quart. Appl. Math., 2, 258-261.

Davidon, W. (1991). Variable Metric Method for Minimization. SIAM J. OPTIMIZATION, 1(1).

Debye, P. (1909). Näherungsformeln für die Zylinderfunktionen für große Werte des Arguments und unbeschränkt veränderliche Werte des Index, Mathematische Annalen, 67(4), 535–558.

Fekedulegn, D., Siurtain, M. M., & Colbert, J. (1999). Parameter estimation of nonlinear growth models in forestry. Silva Fennica, 33(4), 327-336.

Fletcher, R. (1970). A New Approach to Variable Metric Algorithms, Computer Journal, 13(3), 317–322.

Fletcher, R., & Powell, M. (1963). A rapidly convergent descent method for minimization. Comput. J., 6, 163-168.

Fletcher, R., & Reeves, C. (1964). Function minimization by conjugate gradients. The Computer Journal, 7(2), 149-154.

Gauss, C. (1809). Theoria motus corporum coelestium in sectionibus conicis solem ambientium. Dover, New York.

Goldfarb, D. (1970). A Family of Variable Metric Updates Derived by Variational Means, Mathematics of Computation, 24(109), 23–26.

Hestenes, M., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards, 49(6), 409.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics, 2(2), 164-168.

Marquardt, D. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics. 11(2), 431–441.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to linear regression analysis (5th ed.). Hoboken, NJ: Wiley.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. The computer journal, 7(4), 308-313.

Newton, I. (1669). De analysi per aequations numero terminorum infinitas.

Raphson, J. (1697). Analysis aequationum universalis seu ad aequationes algebraicas resolvendas methodus generalis, & expedita, ex nova infinitarum serierum methodo, deducta ac demonstrata. Typis Tho. Braddyll, Prostant Venales Apud Johannem Taylor, Ad Insigne Navis in Coemeterio D. Pauli, 1697.

Seber, G. A., & Wild, C. J. (2003). Nonlinear regression. Hoboken, NJ: John Wiley.

Shanno, David F. (July 1970). Conditioning of quasi-Newton methods for function minimization, Mathematics of Computation, 24(111). 647–656.

Statistical Reference Datasets (StRD). (n.d.). Retrieved from http://www.itl.nist.gov/div898/strd/

Thurber, R., NIST (1979). Semiconductor electron mobility modeling.

Tiku, M. L. & Akkaya, A. D. (2004). Robust estimation and hypothesis testing. New Delhi: New Age International (formerly Wiley Eastern).

Tiku, M. L. Akkaya, A. D. (2010). Estimation in Multifactor Polynomial Regression under Non- normality . Pak. J. Statistics, 25, 49-68.

Tjørve KCM, Tjørve E. Modelling avian growth with the Unified-Richards: As exemplifed by wader-chick growth. Journal of Avian Biology. 2017.

APPENDIX

Simulation Study Codes in R for Thurber Model

```
#required packages
library(nlmrt)
library(minpack.lm)
library(optimx)
library(NISTnls)
library(Bhat)

n=250 #number of observations
M=10000  #number of trials
parm=7 #number of parameters in the model

Mse=NULL
Cf=matrix(ncol=parm , nrow=M)
Var=matrix(ncol=parm , nrow=M)
iter=NULL

Mselm=NULL
Cflm=matrix(ncol=parm , nrow=M)
Varlm=matrix(ncol=parm , nrow=M)
iterlm=NULL
```

```
Msegs=NULL
Cfgs=matrix(ncol=parm , nrow=M)
Vargs=matrix(ncol=parm , nrow=M)
itergs=NULL

Msenm=NULL
Cfnm=matrix(ncol=parm , nrow=M)
Varnm=matrix(ncol=parm , nrow=M)
iternm=NULL

Msecg=NULL
Cfcg=matrix(ncol=parm , nrow=M)
Varcg=matrix(ncol=parm , nrow=M)
itercg=NULL

modelfun=function(b,x) {
(b[1] + b[2]*x + b[3]*x^2 + b[4]*x^3) / (1 + b[5]*x + b[6]*x^2 + b[7]*x^3)
}

#starting value specification
startpar=c(1200,1400,500,65,0.7,0.3,0.03)

for (i in 1:M) {
#generating random data
pred=runif(n,-3.6,2)
#error generation
err=rnorm(n,0,1)
```

```
#true parameter values specification

b1t=1288 ; b2t=1491 ; b3t=583 ; b4t=75 ; b5t=0.97 ; b6t=0.40 ; b7t=0.05

resp=(b1t + b2t*pred + b3t*pred^2 + b4t*pred^3) / (1 + b5t*pred + b6t*pred^2 +
b7t*pred^3) +err

simdata=data.frame(resp,pred)


#using Newton-type method

func=function(b) {

sum((resp-((b[1] + b[2]*pred + b[3]*pred^2 + b[4]*pred^3) /

(1 + b[5]*pred + b[6]*pred^2 + b[7]*pred^3)))^2)

}

gnm=nlm(f=func , p=startpar , iterlim=500)

predicgn=modelfun(b=c(gnm$estimate[1:7]),x=pred)

residgn=resp-predicgn

Mse[i]=mean(residgn^2)

Cf[i , ]=c(gnm$estimate[1:7])

iter[i]=gnm$iterations


#using Levenberg-Marquardt method

model2=nlsLM(resp~(b1 + b2*pred + b3*pred^2 + b4*pred^3) / (1 + b5*pred +
b6*pred^2 + b7*pred^3) ,

start=c(b1=startpar[1],b2=startpar[2],b3=startpar[3],b4=startpar[4],b5=startpar[5],b6
=startpar[6],b7=startpar[7]) , data=simdata)

residlm=resp-predict(model2)

Mselm[i]=mean(residlm^2)

Cflm[i , ]=coef(model2)

iterlm[i]=model2$conv$finIter
```

```
#using BFGS method
func=function(b) {
sum((resp-((b[1] + b[2]*pred + b[3]*pred^2 + b[4]*pred^3) /
(1 + b[5]*pred + b[6]*pred^2 + b[7]*pred^3)))^2)
}
smm=optimx(fn=func , par=startpar , method="BFGS")
predicbfgs=modelfun(b=c(smm$p1,smm$p2,smm$p3,smm$p4,smm$p5,smm$p6,sm
m$p7),x=pred)
residgs=resp-predicbfgs
Msegs[i]=mean(residgs^2)
Cfgs[i , ]=c(smm$p1,smm$p2,smm$p3,smm$p4,smm$p5,smm$p6,smm$p7)
itergs[i]=smm$gevals

#using Nelder-Mead method
func=function(b) {
sum((resp-((b[1] + b[2]*pred + b[3]*pred^2 + b[4]*pred^3) /
(1 + b[5]*pred + b[6]*pred^2 + b[7]*pred^3)))^2)
}
nmm=optimx(fn=func , par=startpar , method="Nelder-Mead")
predicnm=modelfun(b=c(nmm$p1,nmm$p2,nmm$p3,nmm$p4,nmm$p5,nmm$p6,n
mm$p7),x=pred)
residnm=resp-predicnm
Msenm[i]=mean(residnm^2)
Cfnm[i , ]=c(nmm$p1,nmm$p2,nmm$p3,nmm$p4,nmm$p5,nmm$p6,nmm$p7)
iternm[i]=nmm$fevals
```

```
#using Davidon-Fletcher-Powell method
func=function(b) {
sum((resp-((b[1] + b[2]*pred + b[3]*pred^2 + b[4]*pred^3) /
(1 + b[5]*pred + b[6]*pred^2 + b[7]*pred^3)))^2)
}
x <- list(label=c("b1","b2","b3","b4","b5","b6","b7"),
est=startpar ,low=c(0,0,0,0,0,0,0),upp=c(2000,2000,1000,100,5,5,5))
modeldfp=ezgi(x, f=func )
residdav=resp-modelfun(b=modeldfp$est,x=pred)
Msedav[i]=mean(residdav^2)
Cfdav[i , ]=modeldfp$est
iterdav[i]=modeldfp$iter

}

#Newton-type results
#means of estimators
meangsb=NULL
for (i in 1:parm) {
meangsb[i]=mean(Cf[,i],na.rm=TRUE)
}
meangsb
#simulated variances of estimates
varsimgn=NULL
for (i in 1:parm) {
varsimgn[i]=var(Cf[,i],na.rm=TRUE)
}
varsimgn
```

```
#MSE value for the fit
mean(Mse)
#bias calculation for Gauss-Newton
truepar=c(1288,1491,583,75,0.97,0.40,0.05)
biasg=NULL
for (i in 1:parm) {
biasg[i]=mean(Cf[,i],na.rm=TRUE)-truepar[i]
}
biasg
#Mses of the estimates
Msegsb=NULL
for (i in 1:parm) {
Msegsb[i]=varsimgn[i]+biasg[i]^2
}
Msegsb
#number of iterations
mean(iter,na.rm=TRUE)

#Levenberg-Marquardt results
#means of estimators
meanlmb=NULL
for (i in 1:parm) {
meanlmb[i]=mean(Cflm[,i])
}
meanlmb
#simulated variances of estimates
varsimlm=NULL
```

```
for (i in 1:parm) {
varsimlm[i]=var(Cflm[,i])
}
varsimlm
#MSE values
mean(Mselm)
#bias calculation for L-M method
biaslm=NULL
for (i in 1:parm) {
biaslm[i]=mean(Cflm[,i])-truepar[i]
}
biaslm
#Mses of the estimates
Mselmb=NULL
for (i in 1:parm) {
Mselmb[i]=varsimlm[i]+biaslm[i]^2
}
Mselmb
#number of iterations
mean(iterlm,na.rm=TRUE)

#BFGS results
#means of estimators
meanbfgs=NULL
for (i in 1:parm) {
meanbfgs[i]=mean(Cfgs[,i])
}
meanbfgs
```

```
#simulated variances of estimates
varsimbfgs=NULL
for (i in 1:parm) {
varsimbfgs[i]=var(Cfgs[,i])
}
varsimbfgs
#MSE values
mean(Msegs)
#bias calculation for L-M method
biasbfgs=NULL
for (i in 1:parm) {
biasbfgs[i]=mean(Cfgs[,i])-truepar[i]
}
biasbfgs
#Mses of the estimates
Msebfgs=NULL
for (i in 1:parm) {
Msebfgs[i]=varsimbfgs[i]+biasbfgs[i]^2
}
Msebfgs
#number of iterations
mean(itergs,na.rm=TRUE)

#Nelder-Mead results
#means of estimators
meannm=NULL
for (i in 1:parm) {
meannm[i]=mean(Cfnm[,i])
}
```

```
meannm
#simulated variances of estimates
varsimnm=NULL
for (i in 1:parm) {
varsimnm[i]=var(Cfnm[,i])
}
varsimnm
#MSE values
mean(Msenm)
#bias calculation for Nelder-Mead
biasnm=NULL
for (i in 1:parm) {
biasnm[i]=mean(Cfnm[,i])-truepar[i]
}
biasnm
#Mses of the estimates
Msebnm=NULL
for (i in 1:parm) {
Msebnm[i]=varsimnm[i]+biasnm[i]^2
}
Msebnm
#number of iterations
mean(iternm,na.rm=TRUE)

#DFP results
#means of estimators
meandfp=NULL
for (i in 1:parm) {
meandfp[i]=mean(Cfdav[,i])
```

```
}
meandfp
#simulated variances of estimates
varsimdfp=NULL
for (i in 1:parm) {
varsimdfp[i]=var(Cfdav[,i])
}
varsimdfp
#bias calculation for DFP method
biasdfp=NULL
truepar=c(1288,1491,583,75,0.97,0.40,0.05)
for (i in 1:parm) {
biasdfp[i]=mean(Cfdav[,i])-truepar[i]
}
biasdfp
#Mses of the estimates
Msedfp=NULL
for (i in 1:parm) {
Msedfp[i]=varsimdfp[i]+biasdfp[i]^2
}
Msedfp
#number of iterations
mean(iterdav,na.rm=TRUE)
#MSE(fit)
mean(Msedav)
```