A NOVEL STOPPING CRITERION

FOR OPTIMIZATION

By

VENKATRAM PADMANABHAN

Bachelor of Technology

Osmania University

Hyderabad, India

2003

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2005

A NOVEL STOPPING CRITERION

FOR OPTIMIZATION

Thesis Approved:

Dr. Russell Rhinehart

Thesis Adviser

Dr. Karen High

Dr. Manjunath Kamath

A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to place on record my gratitude and offer sincere thanks to my Advisor Dr. R. Russell Rhinehart, who amply provided guidance, periodical suggestions and encouragement in my Project work. I am indebted to him for the financial support he extended during my MS Program at the Oklahoma State University. I will cherish the memories of the days I was associated with Dr. Rhinehart from whom I have learnt the problem solving approach and scientific research culture. I am sure that my days with Dr. Rhinehart will help me to go forward in my future programs also with conviction and confidence.

I would also like to thank Dr. Karen High and Dr. Manjunath Kamath for all the unhesitating help I have received from them in completing my thesis work. I would like to acknowledge with thanks the help rendered by the faculty and staff of the School of Chemical Engineering at OSU and also the graduate students who have helped me in a number of ways.

Last but not the least, I reminisce fondly the constant encouragement and continuing support extended to me by my family.

TABLE OF CONTENTS

Chapter Page

Chapter Page

Chapter Page

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure Page

NOMENCLATURE

$f_i(x)$	-	function value at a point x
$J_i(x)$	-	Jacobian of $f_i(x)$
I	-	*Identity Matrix*
n	-	order of the matrix
μ_k	-	scalar quantity
p_k	-	step length
$J_i^T(x)$	-	Transpose of the Jacobian
x_k	-	current value of x
x_{k+1}	-	next value of x
x^*	-	optimum value of x for which $f(x)$ is minimum or maximum
X_i	-	process variable
X_{i-1}	-	previous process variable
i	-	time sampling index
$v^2_{f,i}$	-	filtered value of a measure of variance
$v^2_{f,i-1}$	-	previous filtered value
$\delta^2_{f,i}$	-	filtered value of a measure of variance
$\delta^2_{f,i-1}$	-	previous filtered value
N	-	total number of data points
r_A	=	rate of reaction (mol/s)

k_a	=	rate constant (1/s), dependent on temperature
k_b	=	rate constant (1/s), not dependent on temperature
C_A	=	concentration of methyl acetate in feed solution (mol/s)
A	=	frequency factor (1/s)
E	=	activation energy (J/mol)
R	=	gas constant (J/mol K)
T	=	reactor temperature (K)

CHARTER 1

INTRODUCTION

Optimization is one of the oldest branches of mathematics, serving as a catalyst for the development of geometry and differential calculus. Today it finds applications in most of the scientific and engineering disciplines. The importance of optimization lies in its natural occurrence in two fundamental areas of human interest – the physical and social sciences [1, 2] where optimum principles have proved to be fundamental to successful modeling and interpretation of natural phenomenon. Optimization is aimed towards maximizing or minimizing a measure of quality called the objective function. The objective function value depends on the values chosen for the independent variables which are termed as the decision variables and optimization seeks to find the values for the decision variables which result in the best (minimum or maximum) value for the objective function [3]. Optimization in a manufacturing process serves as a very good example for commercial optimization application. Variables such as cost and quantity of the raw materials are optimized to obtain a product of minimum cost or of better quality or both. In this case, the cost and quantity of the raw materials are the decision variables and the cost, quality and quantity of the product are the objective functions. The concept of optimization is explained by a simple example,

Example 1.1: To minimize the function

$$f(x) = (x + 5)^2 \tag{1.1}$$

The function '*f*' to be minimized is called the objective function and the variable '*x*' is the decision variable. The above function can be plotted for different values of '*x*' as shown in Figure 1.1. The optimum for this function occurs at $x = -5$, when the objective function attains the minimum value of 0.

1.1 Minimizing Process Cost

In an industrial process, for example, the criterion for optimum operation is in the form of minimum cost, where the product cost can depend on large number of interrelated variables, termed as decision variables (DV). In mathematics the performance criterion could be, for example, to minimize the integral of the squared difference between a specified function and an approximation to it generated as a function of the controlled parameters. Both of these examples have in common the requirement that a single quantity is to be minimized by variation of a number of controlled parameters.

The importance of process optimization lies not in trying to find out all the factors affecting a system but in finding out, with the least possible effort, the best way to adjust the system to make it run at its best [4]. If this is carried out well, systems can have a more economic and improved design so that they can be operated with more accuracy or

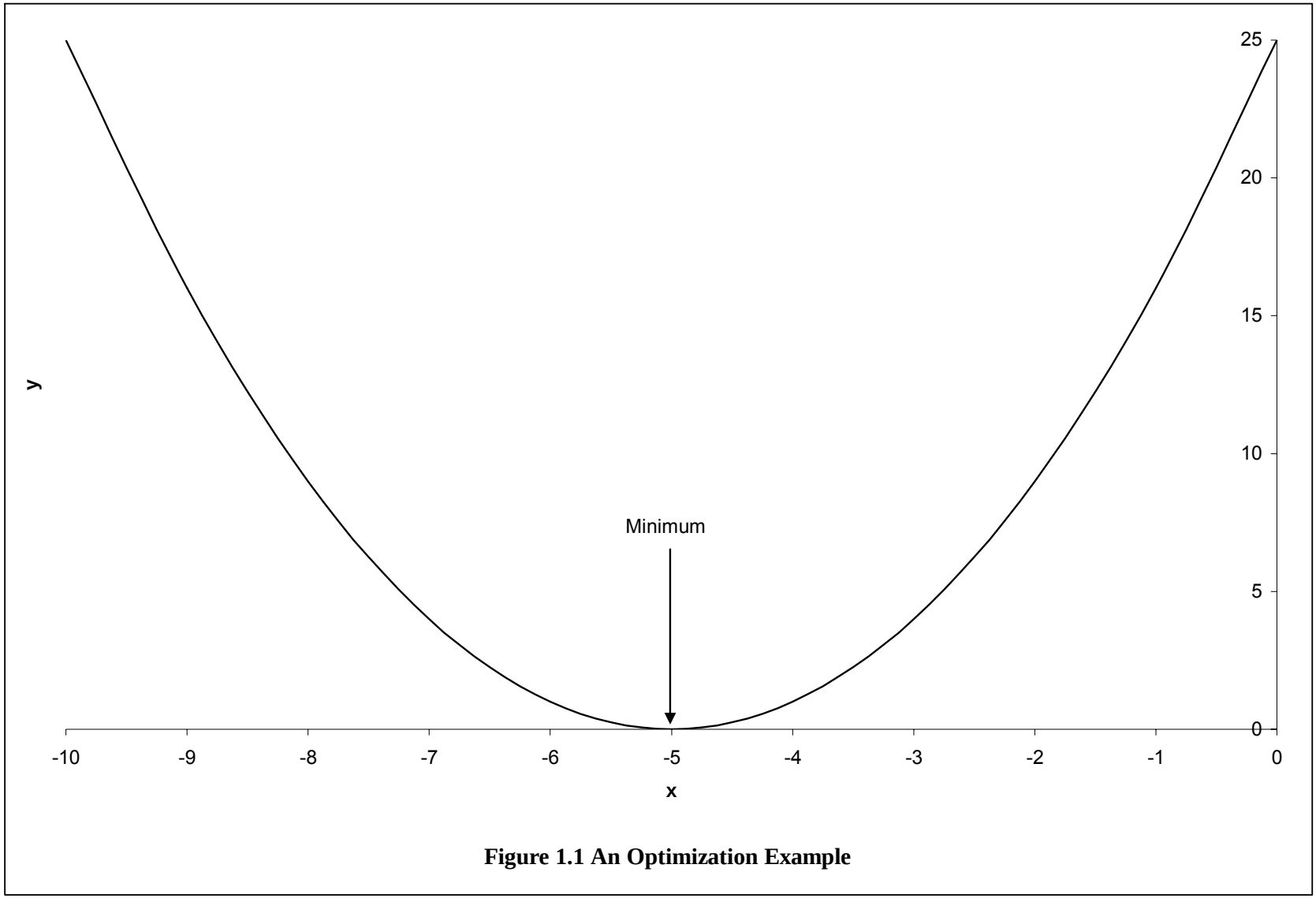


Figure 1.1 An Optimization Example

at less cost and the system designer will have a better understanding of the effects of parameter interaction and variation on his design.

1.2 Empirical Modeling

In many different fields it is necessary to represent a great number of data points in an easily understandable way. Usually, such data points are dependent on one or more independent variables. If the data points are dependent only on one independent variable, it is possible to plot the data points in Cartesian coordinates, and to draw a curve through them. Then this curve is the graphical representation of the data points. If the data points are dependent on more than one independent variable, it is not so easy to produce a graphical representation for them. In this case it is necessary to look for other possibilities of the representation, for instance a functional form. The functional form is nothing but the best model that fits through the noisy data. A functional form is also of interest if the data points are to be used for computations on a digital computer because it is not necessary to store the data points, which can be a very great number, but only the functional form as a representation for them. Moreover, an easy interpolation between data points is possible with the help of a functional form.

For these reasons we must enter into the question how to obtain such a functional form. Usually, a class of functions is selected, for instance the class of polynomials, exponential functions, or trigonometric functions. If we assume that each term of selected class has a parametric representation; in other words, each term is dependent on the decision variables, then the individual functions are characterized by different values for the

parameters. As the functional form shall be a good substitute for the data points, we must determine the parametric values for that particular function which fits the data points best in the sense of an error criterion. As this function is characterized by certain values for the parameters, which are also called the decision variables, we must select the values for these parameters in an appropriate way. This can be done by optimizing the error criterion with respect to the parameters. The determination of a functional form as representation for the data points in this way is usually called curve fitting [5].

Example 1.2: Determine a functional form for noisy data

Consider a noisy data shown in Figure 1.2. The objective of this problem is to find a functional form that closely represents the data. The chosen model is shown in Equation 1.2.

$$y = ax^2 + bx + c \tag{1.2}$$

Constants, a, b, c, are the parameters that have to determined. These are called "controlled parameters" in modeling, but "decision variables" in optimization [6]. Optimization of these parameters based on an error criterion which is, the sum of squared distances between the data points and the respective points on the model curve, shows that the appropriate values for the parameters are

$$a = 1$$
$$b = 2$$
$$c = 3$$

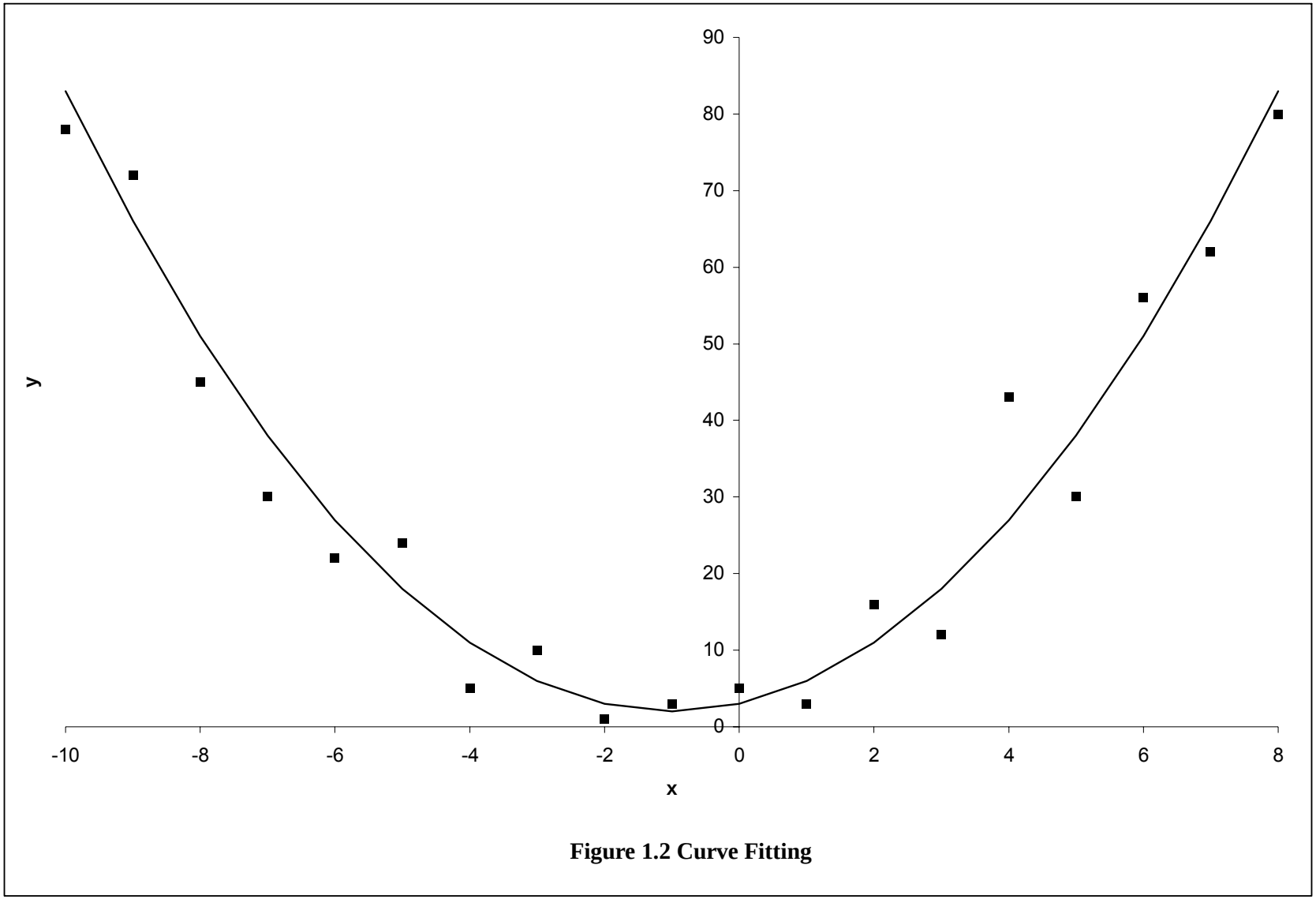


Figure 1.2 Curve Fitting

The model curve that is obtained using these parametric values best fits the noisy data. The requirement of methods of optimization arises from the mathematical complexity necessary to describe the theory of systems, process, equipment and devices which occur in practice. Even quite simple systems must sometimes be represented by theory which may contain approximations, by parameters which change with time, or by parameters that vary in a random manner. For many reasons the theory is imperfect, yet it must be used to predict the optimum operating conditions of a system such that some performance criterion such as low cost or better quality for instance, is satisfied. At best, such theory can predict only that the system is closer to the desired optimum. Optimization methods are then used to explore the local region of operation and predict the way that the system parameters, for example, quantity of the raw materials, should be adjusted to bring the system to an optimum.

1.3 Optimization Categories

There are two main categories in which optimization can be classified. One is constrained optimization and the other is unconstrained optimization. The controlled parameters, which are the decision variables for a particular process, are to be optimized using one of the two main classes of optimization. The constrained optimization tends to seek the optimum values for these parameters in a restricted region where there is a maximum probability of the optimum existing within it. But, in practical situations, we cannot always predict the location of the optimum. In such cases, the optimum values are archived using the unconstrained optimization. There are no bound regions specified for the parameters and optimization algorithm searches for the appropriate values in the

entire range of -∞ to +∞. These two classes of optimization are mainly used in practice to attain economic benefits and empirical modeling. For example, optimization of a set of process setpoints seeking to minimize process operating cost falls under the former case, and optimization of model parameters to fit experimental data is generally called empirical modeling. This work mainly deals with the numerical empirical model optimization of parameters resulting in a functional form that closely represents the noisy experimental data.

The model parameters are usually optimized based on the difference between the predicted value and the experimental value. The sum of squared deviations (SSD) of the data to model is called the error function. The optimization algorithm seeks the optimum values for the parameters by minimizing the error function.

Nonlinear, least squares optimization is commonly used to determine model parameter values that best fits the empirical data, by minimizing the sum of squared deviations (SSD) of data to model, termed the Objective Function (OF). Such models are commonly used in control and optimization. Common multivariable nonlinear optimization methods include Marquardt-Levenberg, Gauss-Newton, Nelder-Mead Simplex, and successive quadratic. Nonlinear optimization proceeds in successive iterations as the search progressively seeks the optimum parameter values, termed decision variables (DV) [7].

As the optimum is approached, the optimization procedure needs a criterion to stop the iterations. The criterion should desirably stop the search when subsequent changes in the

DV values do not improve the OF value. Thus, every optimization algorithm should include a stopping-criterion that stops the process when appropriate values of the parameters are achieved.

Some of the current stop-optimization criteria include [8]

1. A threshold on objective function value, which terminates the optimization process when the objective function value is less than the set value.
2. A threshold on change in the objective function value, which terminates the optimization process when it observes no change in the objective function value.
3. A threshold on change in the decision variable is another widely used criterion, which terminates the process when it observes no change in desired parameter values.
4. A threshold in the number of iterations, which terminates the optimization after carrying out a certain number of iterations irrespective of whether the desired values for the parameters are achieved.
5. Rise in Sum of Squared Deviation (SSD) or Root Mean Square (RMS) for validation set.

Setting up thresholds on any of these factors requires an approximate knowledge of the optimum even before the optimization procedure is carried out. This is important because, if the threshold is set way away from the optimum, there is a possibility of the optimization procedure to stop searching well before the optimum is attained. On the

other hand, if the threshold is set far below the optimum, the optimizer might never find the optimum. Hence, stopping criteria 1-4 require *a priori* knowledge of the appropriate values. They are scale dependent, application dependent, starting point dependent, and optimization algorithm dependent; right choices require human supervision [9]. While criterion 5 has an advantage. It does not require *a priori* knowledge of the optimum. However, it has certain disadvantages attached to it. It stops when the optimizer observes a rise in the SSD value which has a very low probability of occurring.

This work explains, demonstrates, and evaluates a novel stop-iteration criterion for least squares optimization, which is scale-free and requires no prior knowledge of the optimum. It stops iterations when there is no statistical evidence of improvement in successive iterations relative to the variation in the data.

CHAPTER 2

FOCUS ON CURRENTLY USED CRITERIA

There are many features that contribute to the degree of difficulty of an optimization problem. As the wide applicability and the great flexibility of the optimization in industries make it tempting to formulate models with ever increasing numbers of variables, it becomes more difficult to obtain optimum values for all the parameters in the model. Such a problem can be eliminated by using a good optimization algorithm and a proper stopping criterion.

A general algorithm for optimization procedure consists of three major steps: a sampling step, an optimization step, and a check of some optimization stopping criterion. The availability of a suitable stopping criterion is an important aspect of any optimization process.

To minimize computational burden and calculation time, the criterion should be loose enough that it does not require too many function evaluations after the near-optimum point has been found. But to ensure that a good model is obtained, it should also be stringent enough to ensure that in typical cases, the algorithm does not terminate before

the optimum values have been attained, i.e. if the final values obtained are no where near to the optima.

There are a variety of stopping criteria used in the industrial optimization problems. The most commonly used criterion is setting up a threshold value on the objective function. This criterion involves fixing a previously known value for the objective function before the optimization process is started. When the optimization procedure is carried out, the objective function value is evaluated once after each iteration, and is then compared to the previously set threshold. The criterion stops the procedure if the evaluated objective function value is less than or equal to the threshold value. The accuracy of the optimum values for the parameters in the model is dependent on the selection of the threshold objective function value [10]. For example, optimization of a polynomial function to determine its minimum value requires this kind of stopping criteria to be incorporated into the algorithm. Let us consider a polynomial equation with two independent variables, 'x' and 'y'. The values of 'x' and 'y' for which the polynomial function value attains minimum are its optimum values. So, in this case, we can set a threshold value for the polynomial function, which is our objective function, to a number close to its minimum.

The optimization algorithm tends to search for values of the variables, 'x' and 'y', such that the function value approaches the minimum. If the threshold value set is not very close to the minimum, the "optimum" obtained by the optimization would be less accurate. Hence, the values of 'x' and 'y' depend greatly on the previously set threshold value.

In experimental optimization it is usually decided heuristically when to terminate the series of trials; for example when the trial results indicate that no further significant improvement can be gained. In numerical optimization, if the calculations are made by computer, one must build into the program when the optimization procedure is to be terminated. For this purpose, quantitative criteria are needed which refer to the data available at any time. Sometimes, although not always, one will be concerned to obtain a solution as exactly as possible, i.e. accurate to the last stored digit. This requirement can relate to the variables or to the objective function. This criterion for stopping optimization looks at two or more successive values of the decision variables or the objective function. The optimization process is terminated when the criterion observes a change in these values which is less than some threshold. For instance, if we consider the same example of finding the minimum of the polynomial equation as we did earlier, the algorithm tends to take steps toward the optimum values of 'x' and 'y' at every iteration and compare them with the values obtained from the previous iteration. The program exits when it finds no significant improvement in these values, which are called the 'step lengths'.

This procedure has however one disadvantage which can be serious. Small step lengths occur not only when the optimum is nearby, but also if the search is moving through a narrow valley. The optimization may then be broken off long before the sought for extreme value is reached.

The probability that the optimizer attains the optimum values for the variables depends greatly on the initial guesses made to start the procedure. If the initial guess for the variables is no where near to the optimum, the optimizer takes a long time to get to the appropriate values. In such cases, it is convenient to stop the optimization process and rerun it with new initial guesses. Hence, it is required to fix a maximum number of iterations that should be allowed to be carried out by the optimizer to attain the optimum values. Once the maximum number of iterations is reached, the optimizer stops the search and starts again with new set of initial values for the variables.

The different kinds of stopping criteria for optimization mentioned above are scale dependent, application dependent, starting point dependent, and optimization algorithm dependent; right choices require human supervision. However, when evaluating optimization algorithms, the use of *a priori* known information about the objective function under consideration should be refrained from. For example, in a practical situation where there is a need to optimize a process model to obtain the variables associated with it, the threshold value for the objective function (process model) is not available before hand. In such cases, it becomes highly problematic setting up a right threshold value. For instance, consider the simple examples shown below.

Example 2.1: Minimize the function

$$f(x) = x^2 - 2x - 10 \tag{2.1}$$

The optimum for this function, *f* occurs at $x = 1$, when the objective function attains the minimum value of -11. This is clearly shown in Figure 2.1. Obtaining the optimum value for this objective function is not possible unless the user has *a priori* knowledge of it. In this example the optimum could be obtained by using the derivative information. This might not be possible in all the practical cases. If the user sets a threshold value for the objective function close to zero, the optimizer carries out the optimization process and stops when the curve cuts the *x-axis* and return the output as $x = 4.31$ or $x = -2.31$. In this case, the optimizer returned the roots of the polynomial equation and not the optimum.

Example 2.2: Minimize the function

$$f(x) = \frac{27}{4}\left(\left(\frac{1}{x}\right)^9 - \left(\frac{1}{x}\right)^6\right) \tag{2.2}$$

Figure 2.2 shows the pictorial representation of the above Equation 2.2. It is clearly indicated in the figure that the function value attains minimum when *x* value is close to 1.14. When the optimization algorithm searches for the minimum along the deep valley of the curve, it observes very insignificant change in the decision variables (*x* values). The decision variables at two successive iterations are shown in Figure 2.2. If the optimization algorithm has a stopping criterion based on the threshold on the change in the DV, it stops searching for the minimum before it reaches the bottom of the valley. Hence, the optimum is never obtained if the search is carried out along the steepest side of the valley.

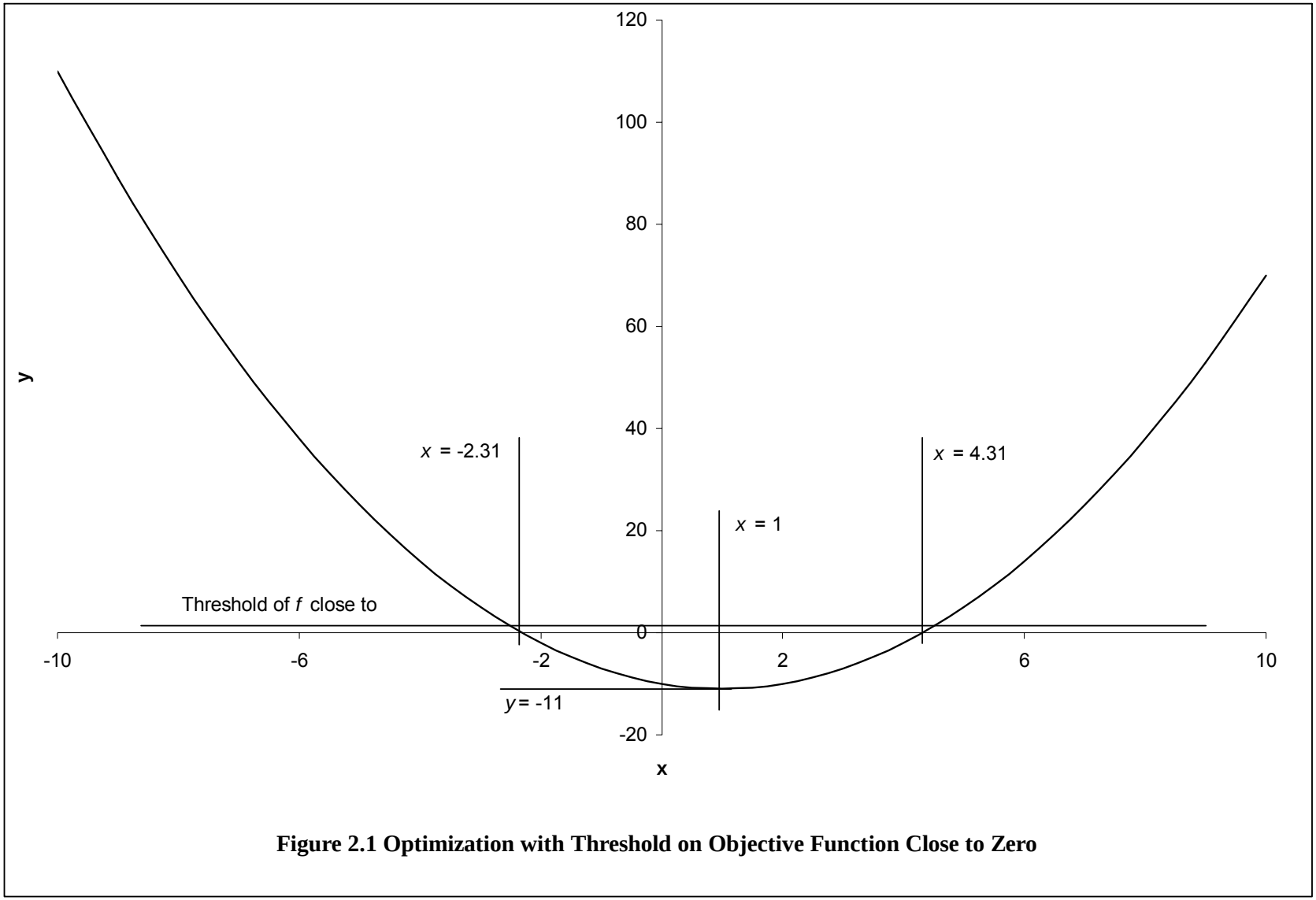


Figure 2.1 Optimization with Threshold on Objective Function Close to Zero

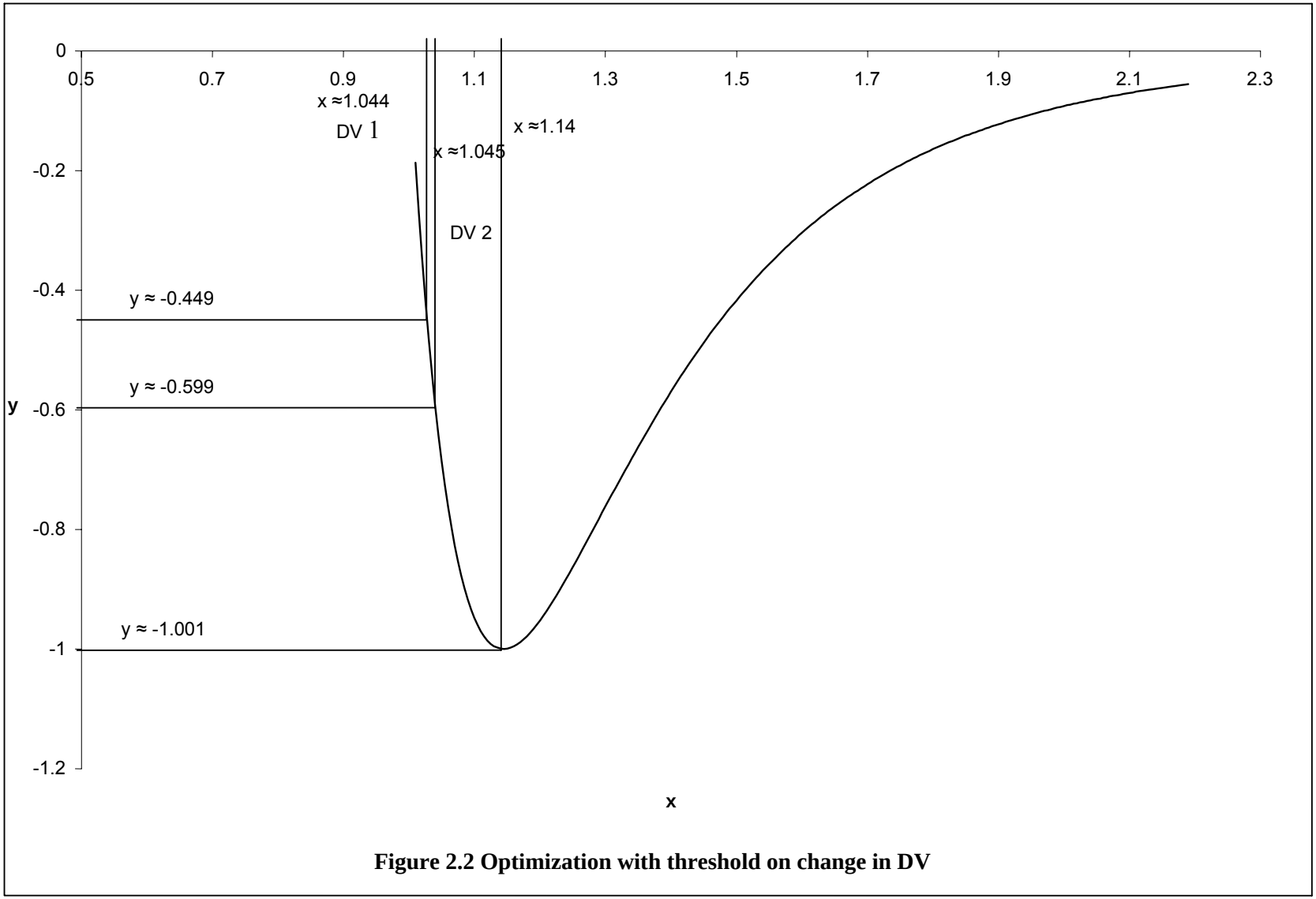


Figure 2.2 Optimization with threshold on change in DV

CHAPTER 3

NOVEL STOPPING CRITERION

An effort has been put in to develop a new stop-optimization criterion to eliminate the various disadvantages of the currently used stopping criteria. This work explains, demonstrates, and evaluates a novel stop-iteration criterion for the least squares optimization, which is scale-free and requires no *a priori* knowledge of the optimum.

The concept of steady state identification technique is used to identify the end point of an optimization process instead of the conventional stopping criteria of setting up thresholds. This identification technique involves the calculation of the sum of squared deviations (SSD) between the data and the model. The optimizer tries to minimize the root mean square of the SSD (RMS SSD) value and the steady state identification technique calculates the ratio of the variances obtained from two different methods and tends to stop the optimization when the ratio statistic is less than unity.

An observer of an optimization procedure for empirical data will note that the RMS SSD between the data and the model, the objective function value (OF) drops to an asymptotic minimum with progressive optimization iterations. The novelty of this method of observing progressive improvement is to calculate the RMS SSD of a random subset

(RMS SSD RS) of data (a different randomly selected subset for each iteration). The RMS SSDRS will appear as a noisy signal relaxing to its noisy steady state value as iterations progress.

By using a random subset of data to provide a RMS SSD value for each iteration, the noise is independently distributed; and, at steady state, when convergence is achieved, the noise reflects the variance in the data. The noise is Chi-Square distributed, with an average equal to the standard error of the residual (model-to-data mismatch). When the noisy signal reaches a statistical steady state, the optimization has progressed to the point where there is no statistically significant improvement in OF with respect to model standard error; and optimization should be stopped. Since, the test looks at signal-to-noise ratio; it is scale independent and "right" for any particular application.

The stopping criterion should be in a position to tell the optimizer that the statistical steady state has been reached and that the optimization process can be stopped. Hence, while developing this novel criterion for stopping optimization, we used the steady state identification technique to predict the end point. There are many ways to determine whether a signal is at steady state, or more properly stated, whether to accept or reject the null hypothesis. The most common technique used is the *ratio of variances*. The ratio of the variances as measured on the same set of data by two different methods is used to identify the steady state. For example, if we have a data of RMS SSD that gradually attains steady state as shown in Figure 3.1, the variances on this set of data are calculated using two different methods as shown below [11].

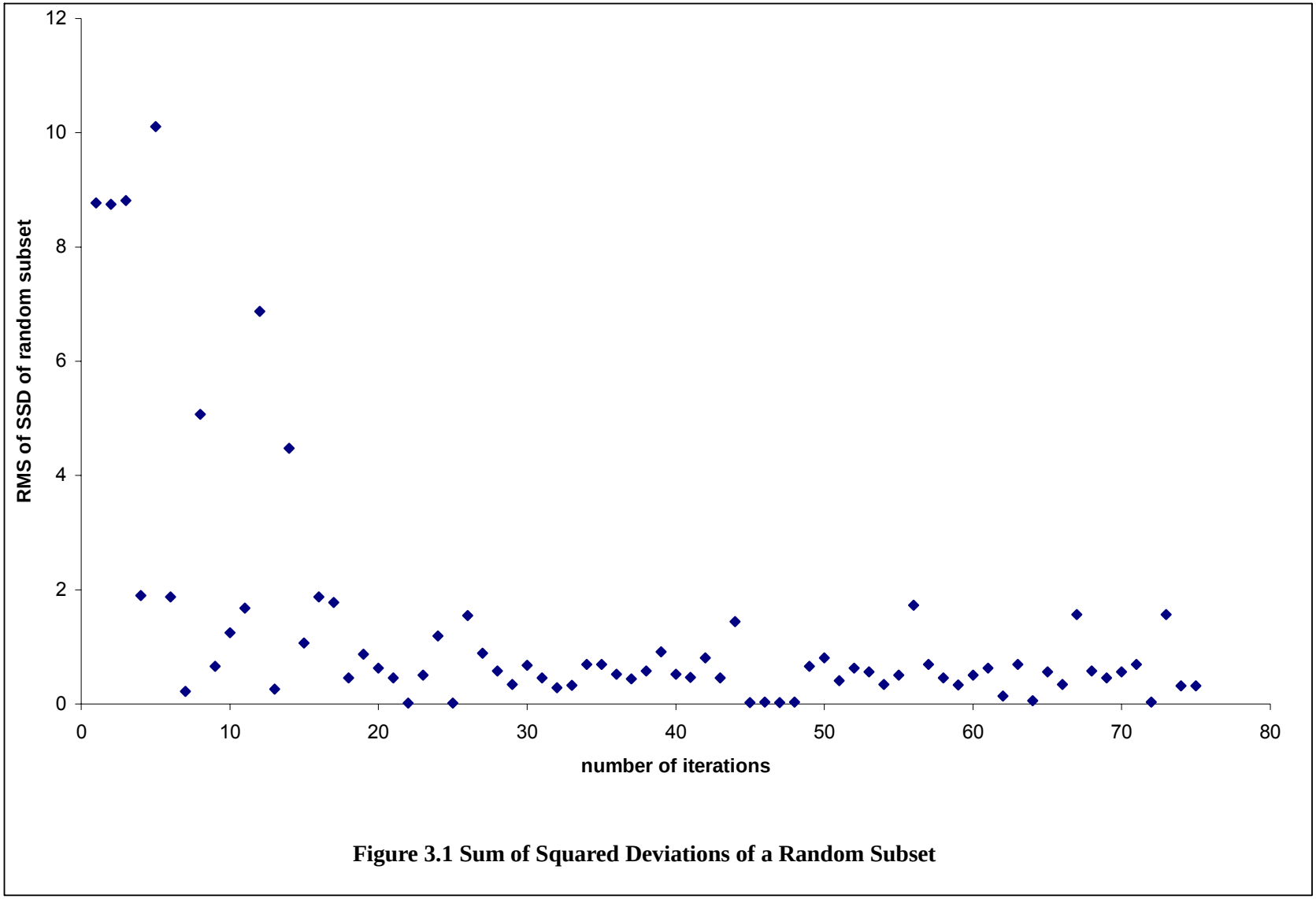


Figure 3.1 Sum of Squared Deviations of a Random Subset

$$Variance1 = \frac{1}{(N-1)}\sum_{i=1}^{N}\left(X_i - \bar{X}_N\right)^2 \tag{3.1}$$

$$Variance2 = \frac{1}{2(N-1)}\sum_{i=1}^{N}\left(X_i - X_{(i-1)}\right)^2 \tag{3.2}$$

Where,

N - total number of data points

X_N - mean value of the data points

X_i - current data point

X_{i-1} - previous data point

Once the variances for this set of data points are obtained, the ratio is evaluated as –

$$Ratio = \frac{Variance1}{Variance2} \tag{3.3}$$

When steady state is reached, the ratio approaches unity.

This method of identifying steady state does not require a pre-defined threshold on the objective function (OF) or *a priori* knowledge of the optimum. However, this method has some disadvantages attached to it. This method of evaluating the variances using the average value is computationally intense and also requires a large storage capacity. To eliminate the computational intensity, we chose the method of Cao and Rhinehart [12]. It presumes no auto-correlation in the noise, a condition which is satisfied by the random selection of data for the objective function value for each iteration. In this method, the

variances are calculated by replacing the average with an exponentially weighted filtered value.

$$v^2{}_{f,i} = \lambda_2 \left(X_i - X_{f_{(i-1)}} \right)^2 + (1 - \lambda_2) v^2{}_{f,(i-1)} \tag{3.4}$$

$$\delta^2{}_{f,i} = \lambda_3 \left(X_i - X_{(i-1)} \right)^2 + (1 - \lambda_3) \delta^2{}_{f,(i-1)} \tag{3.5}$$

The exponentially weighted filtered factor, X_f is defined as

$$X_{f_i} = \lambda_1 X_i + (1 - \lambda_1) X_{f_{(i-1)}} \tag{3.6}$$

Where,

X_i	-	process variable
X_{i-1}	-	previous process variable
i	-	time sampling index
$v^2_{f,i}$	-	filtered value of a measure of variance
$v^2_{f,i-1}$	-	previous filtered value
$\delta^2_{f,i}$	-	filtered value of a measure of variance
$\delta^2_{f,i-1}$	-	previous filtered value

In the above set of equations, λ_1, λ_2 and λ_3 are filter factors. The ratio of the variances is given by

$$R_i = \frac{(2 - \lambda_1) v^2{}_{f,i}}{\delta^2{}_{f,i}} \tag{3.7}$$

Steady state is accepted when the ratio statistic in the method is less than unity.

The criterion detects a transient zone in which the ratio of the variances is greater than 2 and then tends to seek a steady state for the ratio falls less than unity. The logic is shown in Figure 3.2.

There are many advantages in using this technique of identifying steady state as a stopping criterion for optimization. This method does not require human supervision or *a priori* knowledge of the optimum. It is scale independent, computationally simple and requires very low data storage capacity. It also stops the optimization process when there is statistically no evidence of improvement.

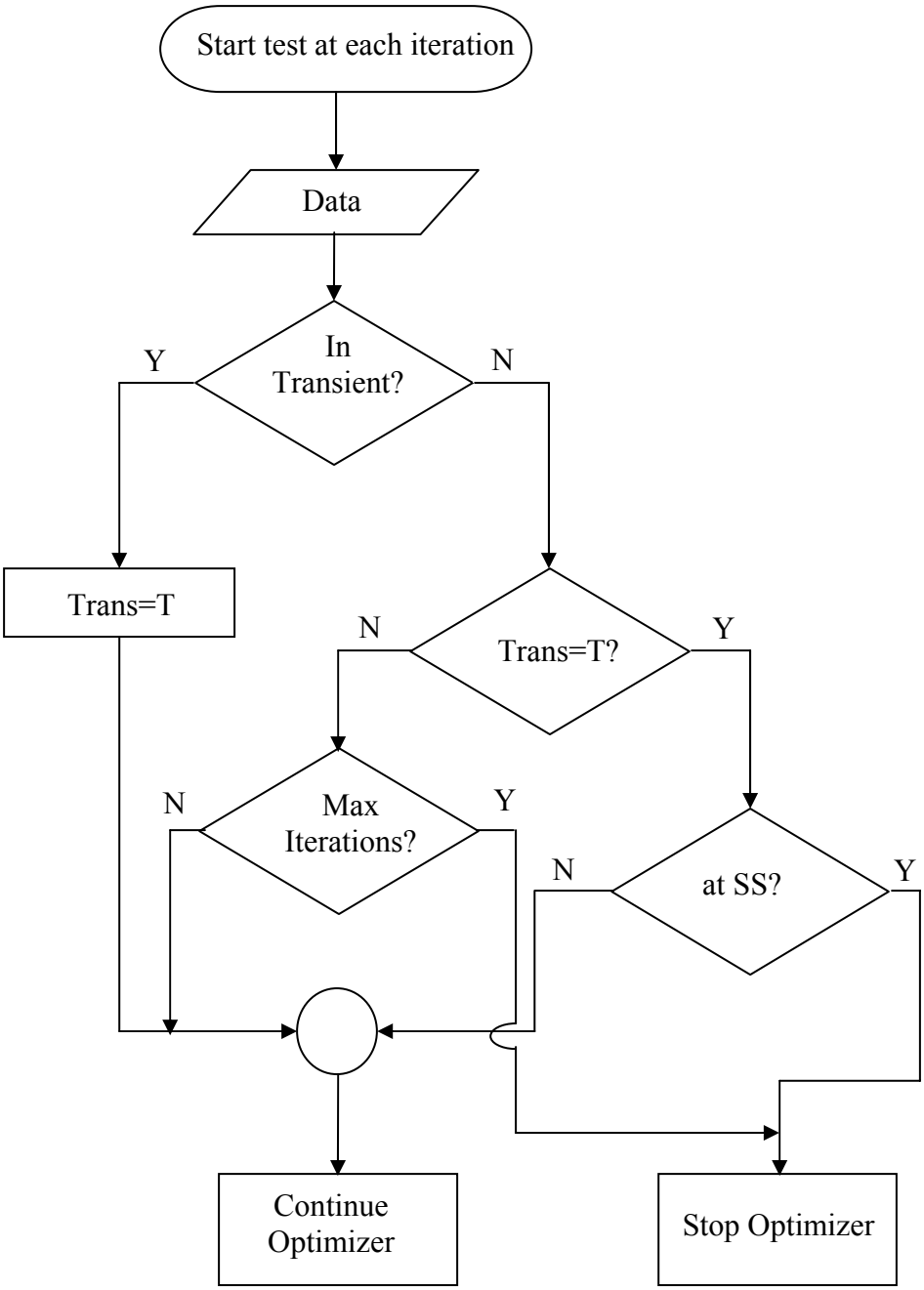


Figure 3.2 Stopping Criteria Logic

CHAPTER 4

PROCEDURE FOR EVALUATION OF THE NOVEL METHOD

The method was examined using three different optimization techniques (Nelder-Mead Simplex, Gauss Newton, and Marquardt-Levenberg) on each of three types of data sets obtained from linear, nonlinear and multivariable functions. For each of the nine cases, the investigation approach is as follows:

1. The optimization methods were run for excessive iterations, as visually defined.
2. After every optimizer iteration, 20% of the total number of data points was randomly selected to calculate the sum of squared deviations.
3. A plot between the root mean square of the sum of squared deviations of the random subset and the number of iterations is made for visual analysis. The method does not require a graph.
4. Model parameter values are recorded twice: first when the random subset of RMS SSD is determined to be at steady state, and finally after excessive iterations.
5. The models that result from these two parameter sets are visually compared by graphs, and quantitatively compared by analysis of variance.

A brief description of the three optimization techniques used to evaluate the novel stopping criterion follows.

4.1 Nelder-Mead Simplex Method

A method that is quite commonly used in nonlinear regression programs is the Nelder-Mead or Simplex method. It is computationally quite simple, other than the calculation of the objective function value. The method works with a number of rules. The starting point is used to construct a simplex, m-dimensional shape with m+1 points, where m is the number of parameters. Thus for a two parameter problem there are three points, a triangle. The program calculates the objective function value at each point of the simplex on the surface [13].

The rules used by the Nelder-Mead Simplex method to approach the minimum are

- Reflect the point with the highest objective function value through centroid (center) of the simplex.
- If this produces the lowest OF value (best point), expand the simplex and reflect further.
- If this is just a good point, start at the top of the simplex and reflect again.
- If this is the highest OF value (worst point), compress the simplex and reflect closer.

These rules are repeated until the convergence criteria are met. The simplex moves over the surface and should contract around the minimum. The simplex method is relatively

robust and numerically less complicated, but it can be inefficient (slow) for simple problems.

For the case of two decision variables, the process generates a sequence of triangles (which might have different shapes), for which the function values at the vertices get smaller and smaller. The size of the triangles is reduced and the coordinates of the minimum point are found.

Let $f(x, y)$ be the function that is to be minimized. To start, we are given three vertices of a triangle $\overrightarrow{V_k} = (x_k, y_k)$, for $k = 1,2,3$. The function $f(x, y)$ is then evaluated at each of the three points $z_k = f(x_k, y_k)$, for $k = 1,2,3$. The subscripts are then reordered so that $z_1 \le z_2 \le z_3$. We use the notation $\vec{B} = (x_1, y_1), \vec{G} = (x_2, y_2), \vec{W} = (x_3, y_3)$ to help remember that $\vec{B}$ is the best vector, $\vec{G}$ is good (next to best), and $\vec{W}$ is the worst vector.

The construction process uses the midpoint $\overrightarrow{M}$ of the line segment joining $\vec{B}$ and $\vec{G}$. It is found by averaging the coordinates

$$\overrightarrow{M} = \frac{1}{2}\left(\vec{B} + \vec{G}\right) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right) \tag{4.1.1}$$

4.1.1 Reflection using the point $\vec{R}$: The function decreases as we move along the side of the triangle from $\vec{W}$ to $\vec{B}$, and it decreases as we move along the side

from $\overrightarrow{W}$ to $\overrightarrow{G}$. Hence it is feasible that $f(x, y)$ takes on smaller values at points that lie away from $\overrightarrow{W}$ on the opposite side of the line between $\vec{B}$ and $\vec{G}$. We choose a test point $\vec{R}$ that is obtained by "reflecting" the triangle through the side $\overrightarrow{BG}$. To determine $\vec{R}$, we first find the midpoint $\overrightarrow{M}$ of the side $\overrightarrow{BG}$. Then draw the line segment from $\overrightarrow{W}$ to $\overrightarrow{M}$ and call its length *d*. This last segment is extended a distance *d* through $\overrightarrow{M}$ to locate the point $\vec{R}$ [13]. The vector formula for $\vec{R}$ is

$$\vec{R} = \overrightarrow{M} + \left(\overrightarrow{M} - \overrightarrow{W}\right) = 2\overrightarrow{M} - \overrightarrow{W} \tag{4.1.2}$$

4.1.2 Expansion using the point $\vec{E}$: If the function value at $\vec{R}$ is smaller than the function value at $\overrightarrow{W}$, and then we have moved in the correct direction toward the minimum. Perhaps the minimum is just a bit farther than the point $\vec{R}$. So we extend the line segment through $\overrightarrow{M}$ and $\vec{R}$ to the point $\vec{E}$. This forms an expanded triangle $\overrightarrow{BGE}$. The point $\vec{E}$ is found by moving an additional distance *d* along the line joining $\overrightarrow{M}$ and $\vec{R}$. If the function value at $\vec{E}$ is less than the function value at $\vec{R}$, then we have found a better vertex than $\vec{R}$. The vector formula for $\vec{E}$ is

$$\vec{E} = \vec{R} + \left(\vec{R} - \overrightarrow{M}\right) = 2\vec{R} - \overrightarrow{M} \tag{4.1.3}$$

4.1.3 Contraction using the point $\vec{C}$: If the function values at $\vec{R}$ and $\overrightarrow{W}$ are the same, another point must be tested. Perhaps the function is smaller at $\overrightarrow{M}$, but we cannot

replace $\overrightarrow{W}$ with $\overrightarrow{M}$ because we must have a triangle. Consider the two midpoints $\overrightarrow{C_1}$ and $\overrightarrow{C_2}$ of the line segments $\overrightarrow{WM}$ and $\overrightarrow{MR}$, respectively. The point with the smaller function value is called $\vec{C}$, and the new triangle is $\overrightarrow{BGC}$.

4.1.4 Shrink toward $\vec{B}$: If the function value at $\vec{C}$ is not less than the value at $\overrightarrow{W}$, the points $\vec{G}$ and $\overrightarrow{W}$ must be shrunk toward $\vec{B}$. The point $\vec{G}$ is replaced with $\overrightarrow{M}$, and $\overrightarrow{W}$ is replaced with $\vec{S}$, which is the midpoint of the line segment joining $\vec{B}$ with $\overrightarrow{W}$ [13].

The search procedure for the Nelder-Mead Simplex method is illustrated in Figure 4.1.

4.2 Gauss-Newton Method

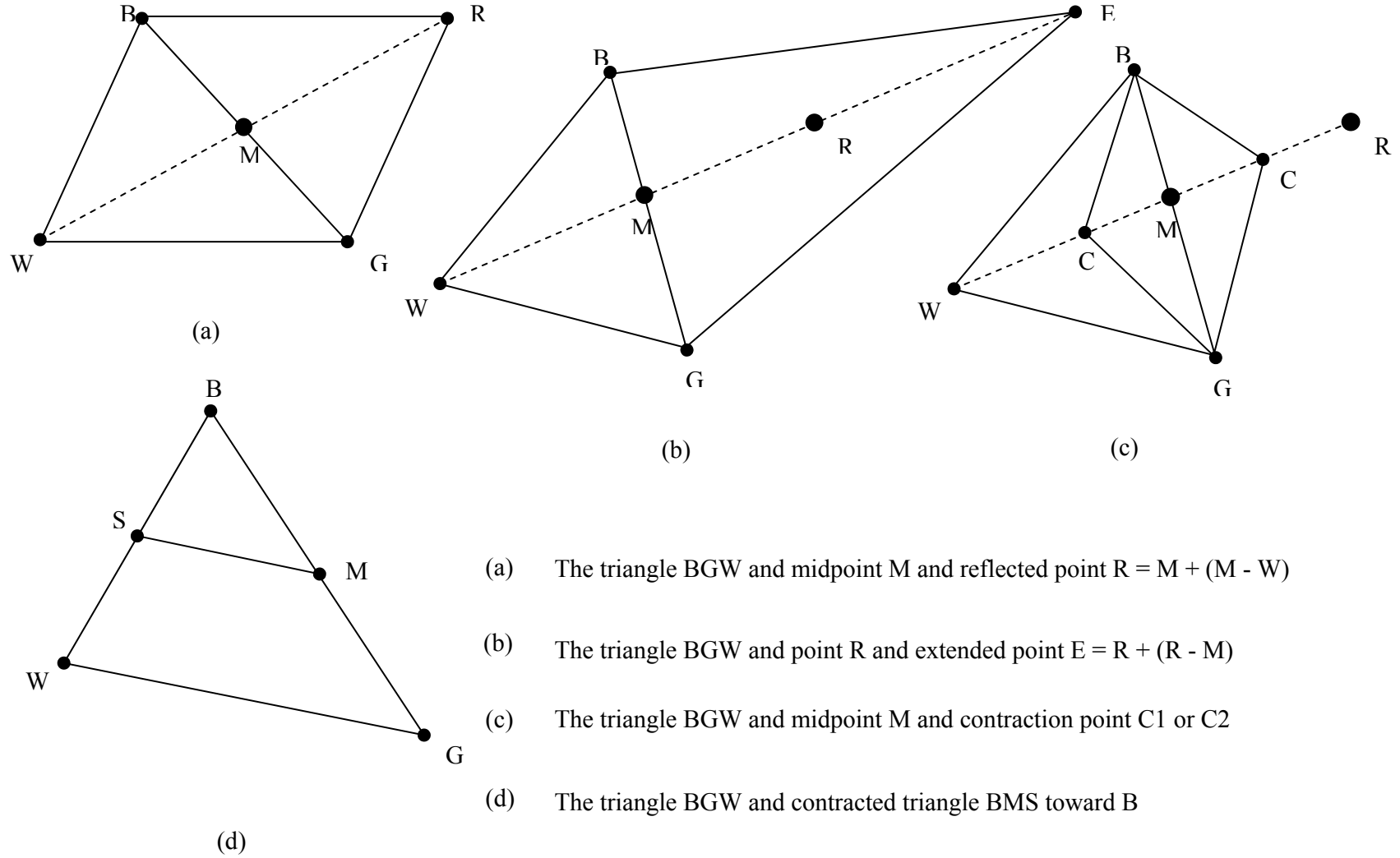
The Gauss-Newton algorithm is used to solve nonlinear least squares problems. It is a modification of Newton's method that does not use second derivatives. The basic iteration of the Newton's Method is given as

$$\left(J_k{}^T J_k + S_k\right)p_k = -J_k{}^T f_k \tag{4.2.1}$$

$$x_{k+1} = x_k + p_k \tag{4.2.2}$$

By neglecting the S_k in Newton's method, Equation (4.2.1) becomes

$$\left(J_k{}^T J_k\right)p_k = -J_k{}^T f_k \tag{4.2.3}$$



(a) The triangle BGW and midpoint M and reflected point R = M + (M - W)

(b) The triangle BGW and point R and extended point E = R + (R - M)

(c) The triangle BGW and midpoint M and contraction point C1 or C2

(d) The triangle BGW and contracted triangle BMS toward B

Figure 4.1 Nelder Mead Simplex Method

this, together with the step (4.2.2), defines the Gauss-Newton method. The simultaneous linear equations (4.2.3) are called the least squares normal equations. Equation (4.2.3) is likely to be less troublesome for the generation of the descent directions than the corresponding one (4.2.1) for Newton's method since the matrix, $J_k^T J_k$ is always at least positive semidefinite. To see why this is, take an arbitrary vector $z \neq 0$ and set $y = J_k z$. Then

$$z^T J_k^T J_k z = y^T y \geq 0 \tag{4.2.4}$$

The only trouble that can arise in this respect is when J_k is rank deficient and hence $J_k^T J_k$ is singular. However, even if p_k is a descent direction this does not guarantee that $F_{k+1} < F_k$. The step (4.2.2) might be too large, locating x_{k+1} at a point well beyond the linear minimum. For these reasons a good starting point is required if there is to be any chance of convergence to a minimum [13].

4.3 Marquardt-Levenberg Method

The Marquardt-Levenberg method is a nonlinear optimization and equation solving technique. The algorithm can be used to estimate unknown variables in sets of nonlinear equations where the number of variables is less than or equal to the number of equations. Simple constraints on the parameters may be used to keep the solution in bounds. The Marquardt-Levenberg method overcomes the drawbacks of the Newton's method by starting off as a direct search algorithm and then progressively becomes gradient-based as the solution converges to the optimum. Marquardt-Levenberg method thus combines the

best features of the gradient Newton-Raphson procedures by using a suitable weighting parameter. The method has the stability of gradient procedure with respect to poor starting values, and at the same time, it possesses the speed of convergence of the Newton-Raphson method when close to the final solution. The main drawback of the Marquardt-Levenberg method is the inability to handle constrained optimization problems [13].

The Marquardt-Levenberg method tries to find the minimum of the function, *f(x)* that is the sum of squares of the nonlinear functions,

$$f(x) = \frac{1}{2}\sum_{i=1}^{m}[f_i(x)]^2 \tag{4.3.1}$$

If the Jacobian of $f_i(x)$ be denoted by $J_i(x)$, then the Marquardt-Levenberg method searches for the minimum in the direction given by the solution '*p*' to the equations

$$\left(J_k{}^T J_k + \mu_k I\right)p_k = -J_k{}^T f_k \tag{4.3.2}$$

where, $\mu_k \geq 0$ is a scalar and *I* is the unit matrix of order n. Equation (4.2.2) is used to obtain a point with which the next iteration is carried out.

$$x_{k+1} = x_k + p_k \tag{4.3.3}$$

For a sufficiently large value of μ_k, the matrix $\left(J_k{}^T + \mu_k I\right)$ is positive definite and p_k is then a descent direction. As $x_k \longrightarrow x^*$, however, we require that $\mu_k \longrightarrow 0$ so that the method acquires the asymptotic rate of convergence of the Gauss-Newton method.

When $\mu_k = 0$, p_k is the Gauss-Newton vector. As $\mu_k \longrightarrow \infty$, the effect of the term $\mu_k I$ increasingly dominates that of $J_k{}^T J_k$ so that, $p_k \longrightarrow -\mu_k{}^{-1} J_k{}^T f_k$ represents an infinitesimal step in the steepest descent direction. Between these two extremes, both p_k and the angle between p_k and $-g_k$ decreases monotonically as μ_k increases [14]. This property is useful because, while the magnitude of the Gauss-Newton vector is a rough indication of an acceptable step length, increasing the bias of p_k towards the steepest descent direction makes p_k more and more likely to be too large a step to give a reduction in function value. The set of all points $x_k + \alpha_k, 0 < \alpha_k \leq 1$, as μ_k varies from 0 to ∞ defines part of a hyperplane in the space of the variables known as a *region of trust* [15].

Each of these three optimization approaches were used to test the novel stopping criterion on three simple but diverse simulated applications and two experimental applications. The simulated applications included the data generated using a linear function, nonlinear function and a multivariable nonlinear function.

4.4 Description of the Functions Used To Generate Data

4.4.1 Linear Function: The model equation selected for this linear problem is $y = Ax + B$ and the number of data points is 20. The linear model that was used to

generate the data is given by $y = A(x + randn) + B + randn(size(x))$. The '*randn*' function adds Gaussian distributed, zero mean, unity variance, random variation [NID (0, 1)] to a particular "*x*" value. Adding uncertainty to the independent variable is a non-conventional practice, but adds realism by simulating uncertainty in experimental control. The "*size(x)*" argument generates a vector of perturbations to the vector of "*y*" values – of the same number of elements as the "*x*" vector.

4.4.2 Nonlinear Function: The model equation selected for this nonlinear problem is $y = A\ln(Bx)$ and the number of data points is 40. The nonlinear model that was used to generate the data is given by $y = A\ln(B(x + randn)) + randn(size(x))$.

4.4.3 Multivariable Function: The model equation selected for this multivariable problem is $z = A\sqrt{x} + B\sqrt{y}$ and the number of data points is 20. The multivariable model that was used to generate the data is given by $z = A\sqrt{(x + randn)} + B\sqrt{(y + randn)} + randn(size(x))$.

The working of the stopping criterion was also validated using two experimentally generated data.

CHAPTER 5

EXPERIMENTAL SETUP

A brief description of the equipment used to obtain the experimental data is given below. The novel stopping criterion for optimization was validated using the experimental data obtained from the two phase flow apparatus and the packed bed reactor.

5.1 Two-Phase Flow Apparatus

The experimental apparatus consists of a vertical pipe through which the air/water mixture flows, a control computer, Camile software, pressure transducers, three orifice meters, each paired with a control valve, piping, two rotameters for airflow (high and low flow rates), one rotameter for water flow and pressure gauges.

Rotameters provide the flow rate information for the air and water streams. These are used in coordination with three orifice meters and the Camile software of the control computer to allow the user to monitor fluid flow rates. The flow rates for both air and water are set to the desired value using the control computer. Real time flow rate values can then be monitored through orifice meters displayed by the control computer or by utilizing the rotameters. Pressure transducers measure the pressure at both the top and the

bottom of the vertical column [14]. The flow diagram of the two-phase flow apparatus is shown in Figure 5.1 (refer Appendix B for experimental data).

5.1.1 Operating Limitations: In order to operate the two-phase flow apparatus effectively, it is imperative to know and understand the limitations of the equipment. The maximum pressure limit of the piping is 120 psig. However, the compression joint will release at approximately 100 psig. To avoid this, the system must be operated such that the air pressure does not exceed 80 psig. The electrical current through all the computer controlled units should be limited to 4-20 mA. Inaccurate readings may result from operations above or below this range. When the air flow rate through the larger pipe is below 1 ft^3/min, there is a possibility that the static head created by the water in the vertical tube is too great for the air to overcome. This could lead to stagnation period where no bubbles appear in the clear tubing even though Camile reports a flow rate. To avoid this problem, the small air pipe should be used when working with air flow rates less than 1 ft^3/min.

5.1.2 Experimental Description: A series of runs was conducted in order to collect the data for calculations. The first step involved in this process was to start-up the Camile TG 4.0 software and perform all the steps needed to run the program (see Appendix A for instructions on the start-up procedures for Camile TG 4.0). The experiment was run using the computer operator, "virtual employee". The "virtual employee" is a macro which runs through the Camile program [16]. Multiple experimental runs can be performed automatically through the use of simple programming within a file. This file can be

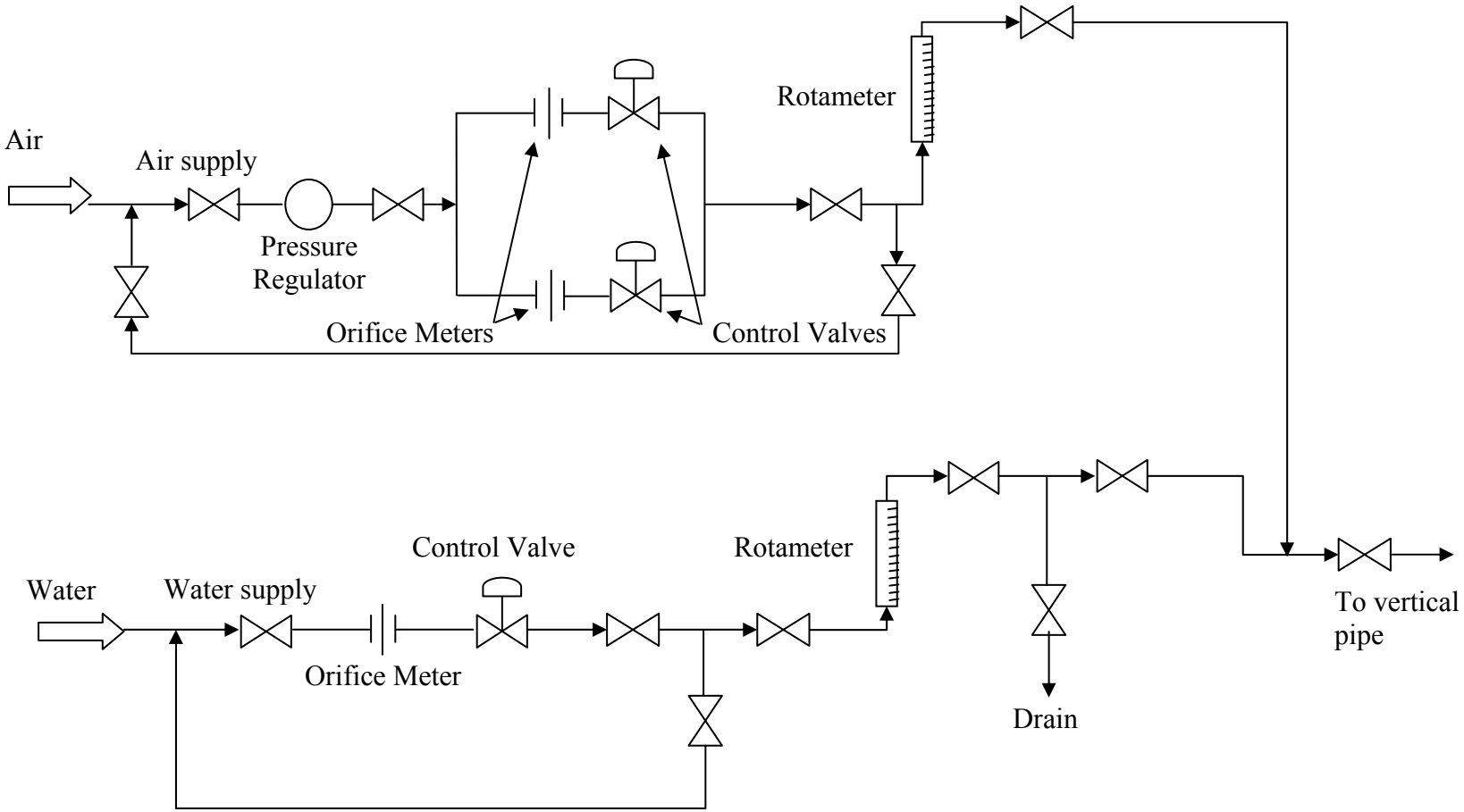


Figure 5.1 Flow Diagram of Two-Phase Flow Apparatus

edited or a new file can be entered. The file consists of a series of lines that contain four digits separated by commas. These digits represent: solenoid valve (1 = open, 0 = closed), large air valve flow rate, small air valve flow rate and water flow rate. Camile uses the information to set the conditions for a particular run. Once steady state is reached, the "virtual employee" goes on to the next line in the experimental plan begins a new run.

The pressure drop within the vertical pipe is found using the recorded flow rates and the water height in the column provided by Camile, the data are used to test the Lockhart-Martinelli model.

5.2 Packed Bed Reactor

The catalytic decomposition of methyl acetate is carried out in a bench-scale model of a catalyzed packed reactor in the Unit Operations Laboratory. The model consists of a feed tank, a pump, a heat exchanger, rotameter, heater and a catalyzed packed bed. The feed tank holds the solution of methyl acetate and the pump propels the solution through the system. The rotameter displays the flow rate of the methyl acetate solution traveling through the system. The methyl acetate solution passes through the tube side of a heat exchanger. The heat exchanger has hot water from the heater flowing on the shell side. The heated methyl acetate solution is then fed into the reactor from the bottom, and a mixture of methanol, acetic acid and methyl acetate exits from the top of the reactor. The experimental setup is shown in Figure 5.2.

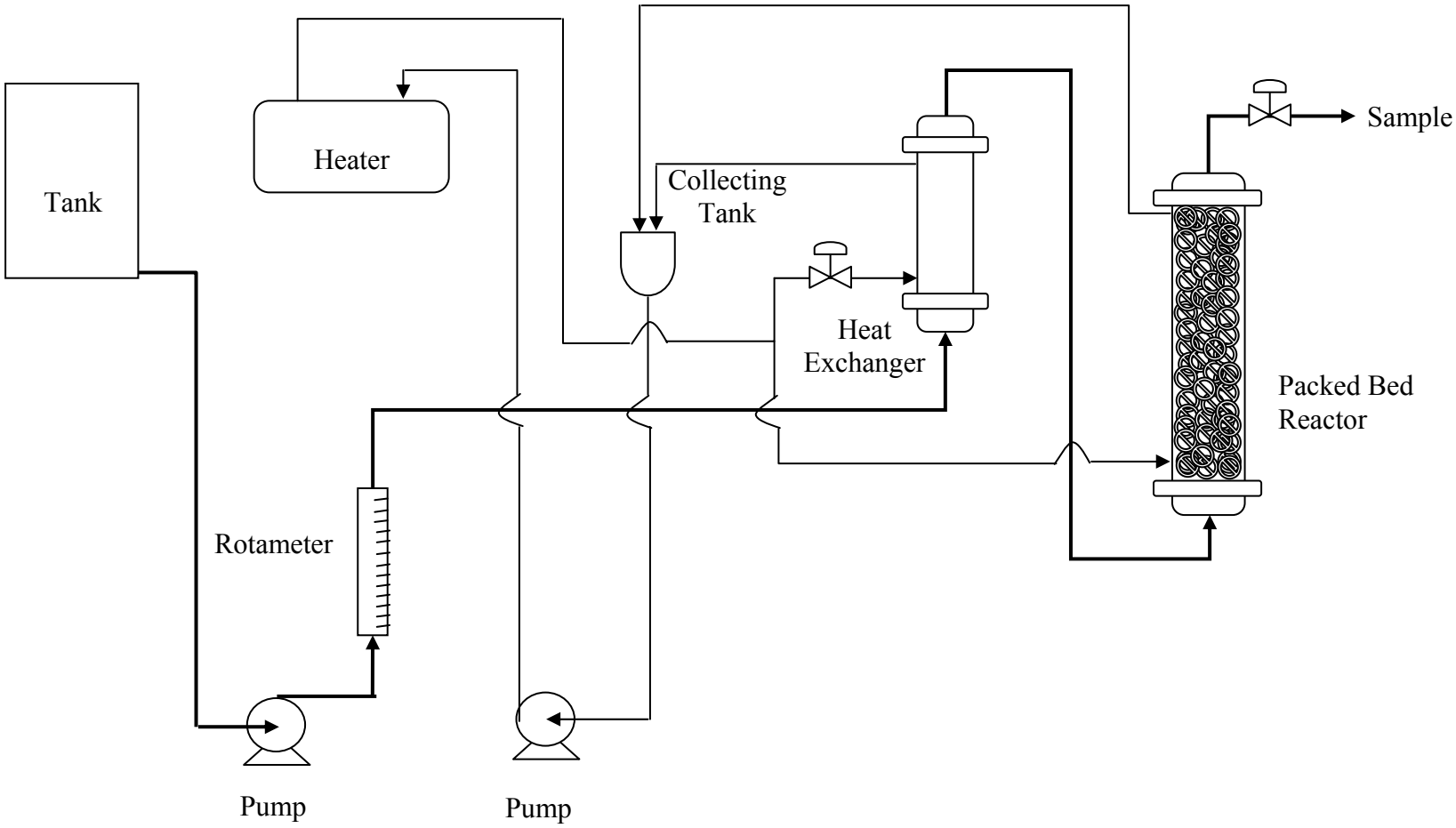


Figure 5.2 Flow Diagram of Packed Bed Reactor

The temperature of the hot water and the effluent stream are measured by a thermocouple. The decomposed mixture is collected in Erlenmeyer flasks and titrated with NaOH solution in burette. The indicator used in titration is phenolphthalein dissolved in ethanol. Phenolphthalein lends the solutions a pink color when the end point of the titration is reached [17]. The methyl acetate solution was prepared by measuring volumes and calculating mass using densities of water and methyl acetate to form an 8% weight solution. The samples were collected using a graduated cylinder with a +/- 0.2 ml error, the collected samples are then put in properly labeled Erlenmeyer flasks for titration. Solutions of methyl acetate and NaOH were made using volumetric glassware and an electric scale for measurements of the latter. The samples were titrated against 0.2 M NaOH, which was dispensed from a burette.

The PBR is filled with a strong acid cation exchange resin catalyst that facilitates the decomposition of methyl acetate into methanol and acetic acid. The reaction takes place on the surface of the catalyst after water and methyl acetate are absorbed. Acetic acid and methanol that results from the reaction are then desorbed.

The decomposition of methyl acetate is given by the following reaction in Equation 6.2.

$$CH_3COOHCH_3 + H_2O \rightarrow CH_3OH + CH_3COOH \qquad (6.2)$$

As the reaction is run in a catalytic packed bed reactor, the reactant must migrate through the packed bed causing axial dispersion. Moreover, the reaction is catalyzed by an ion

exchange resin surface. So, Hougen-Watson kinetics is used to determine a rate expression for the reaction [17].

Using a simplified Hougen-Watson type equation, with water in great excess, the reaction rate is given by Equation 6.3.

$$r_A = \frac{k_a C_A}{1 + k_b C_A} \tag{6.3}$$

Where: r_A = rate of reaction (mol/s)

k_a = rate constant (1/s), dependent on temperature

k_b = rate constant (1/s), not dependent on temperature

C_A = concentration of methyl acetate in feed solution (mol/s)

By the Arrhenius equation, the rate constant as function of temperature is given by Equation 6.4.

$$k_a = A e^{\frac{-E}{RT}} \tag{6.4}$$

Where: A = frequency factor (1/s)

E = activation energy (J/mol)

R = gas constant (J/mol K)

T = reactor temperature (K)

Substituting the above expression for k_a in Equation 6.3, the reaction rate as a function of temperature is give by Equation 6.6.

$$r_A = \frac{Ae^{-\frac{E}{RT}}C_A}{1+k_bC_A} \tag{6.5}$$

The reaction was carried out at a temperature ranging from 25 to 85^{o}C with the initial concentration of methyl acetate equal to 1.0, 1.5 and 2.0 moles/liter at each reaction temperature. The data thus obtained was fed into the optimizer to obtain the optimum values for the parameters such as the frequency factor, A, activation energy, E and the rate constant, k_b. All the three optimization techniques such as the Nelder-Mead Simplex, Marquardt-Levenberg and the Gauss-Newton methods were used to evaluate the novel stopping criterion for optimization. The experimental data are shown in Appendix B.

5.2.1 Safety: Safety is of most priority in an experiment where corrosive materials are being used. Methyl acetate decomposes into methanol and acetic acid. These chemicals have hazards associated with them. As a safety precaution while running this experiment, splash goggles should be worn to prevent eye irritation. Optic nerve is the predominate hazard of chronic exposure to NaOH. Gloves should be used to prevent skin irritation. Personal safety measures should be taken to avoid ingestion and inhalation of these chemicals. Methyl acetate should be mixed in the fume-hood as over exposure affects the lining of the sensitive tissues in the nostrils. All chemicals should be kept away from open flames because methanol and methyl acetate are flammable liquids with

a low flash point [18]. The unit operations stand has an electric power supply for the pump and water heater. Hence, extra care should be taken to avoid wet contact with open power outlets. In case of any spill around the electric power source, the power outlet should be disconnected and left to dry before connecting again.

5.2.2 Environmental Considerations: Acids and hydrocarbons are unsafe to the environment, especially when they contaminate the water supply. Caution should be used when handling and disposing of these chemicals. If released into the soil, the chemicals may leach into groundwater, but are expected to quickly evaporate. Moreover, quick evaporation is expected if chemicals are released into air. The materials are not expected to be toxic to aquatic life.

CHAPTER 6

RESULTS AND DISCUSSION

The results obtained from both the simulated and the experimental data are discussed in this chapter. The simulated data was generated using three different types of models (linear, nonlinear and multivariable nonlinear equations). The experimental data was obtained from packed bed reactor and the two-phase flow apparatus by a group of undergraduate students in the Unit Operations Lab at OSU.

6.1 Results from the Simulated Data

Models of varying complexities were selected to generate the nominal data required to conduct the optimization procedure. In order to make the nominal data representative of an experimental measurement, noise was added to it using a normally distributed random numbers with a variance equal to 1. The generated noisy data was then fed into the optimizer to determine a best-fit empirical model, and the optimization procedure was run for an excessive number of iterations. The parameter values obtained at the end of the optimization process were used to evaluate the values of the objective function and to check if the curve fits the generated noisy data well. The novel stopping criterion was then used to locate a new termination point and the parameter values at that point were again used to evaluate the objective function values and to check if the curve fits the data.

A sample calculation procedure for the novel stop-iteration technique is given in Appendix C. The two curves obtained were compared using the F and p-statistics. The F-statistic is calculated by the ratio of squared residuals, the sum of squared deviations between data and the model based on excessive iterations. The expression for the F-statistic is given by Equation (6.1).

$$F - statistic = \frac{1}{N-1}\frac{\sum(SSD_1)}{\sum(SSD_2)} \tag{6.1}$$

Where, SSD_1 and SSD_2 are the sum of squared deviations of the experimental data from the curves obtained by using the two stopping criterion. The optimization result with excessive iterations is accepted as the most perfect model for the particular random realization of the data. It is expected that any model from fewer iterations should not have as good a SSD, and the F-statistic values should be less than 1.0. However, if the new stopping criterion is good, the ratio of SSD measures will be close to unity.

The p-value indicates the percentiles of the F distribution. It is the one sided probability of obtaining the higher F-value by chance.

The different models and the various optimization techniques used are clearly discussed below.

6.1.1 Optimization of Parameters in a Linear Function

Linear function used: $y = Ax + B$

Parameters to be optimized: A and B

The above mentioned linear function was used to generate the data. The objective function values (*y* values) were calculated for A = 0.5 and B = 0.2 in a range of '*x*' values from 0 to 10 with the interval of 0.5. Gaussian distributed random numbers [NID (0, 1)] were added to the above generated data using the random number generation code in MATLAB 6.5. The noisy data was then used by the optimizer to determine the best empirical values of A and B. The optimization code for different methods to optimize the parameters is written in MATLAB 6.5 release 13 (refer Appendix D). The optimization procedure was run for 60 iterations and the parameter values obtained were recorded to calculate the objective function values. The excessive number of iterations was decided on the basis of change in the sum of squared deviations of the random subset. Another set of parameter values was obtained at a point where the novel stopping criterion suggested termination. The results obtained using the three optimization techniques, *viz*. Nelder-Mead Simplex method, Marquardt-Levenberg method and the Gauss-Newton method are discussed in cases below.

Case 6.1.1.1 Optimization Technique used: Nelder-Mead Simplex

Three random initial values, to form the first simplex, were given to each of the parameters that are to be optimized using the Nelder-Mead Simplex method. The

optimization procedure was then run for an excessive number of iterations until no change in the SSD of the random subset was observed. The plot showing the change in the sum of squared deviations of the random subset with the iterations is shown in Figure 6.1. The number of iterations, took to obtain the optimum values of the parameters using the novel stopping criterion, is clearly indicated in Figure 6.1. The objective function values that resulted from the latter set of parameter values were compared to that obtained from the former using the F and p-statistics. The F and p-statistics and the parameter values for both the curves are shown in Tables 6.1 and 6.2. The comparison plot is shown in Figure 6.2. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

Table 6.1: Goodness of fit for the linear model using Nelder-Mead Simplex method

Test	**Results**
F-Statistic	0.9997
p-Value	0.4990

From Table 6.1, it can be observed that the F-statistic is close to unity and the p value close to 0.5 which suggests that both the curves are statistically indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.2.

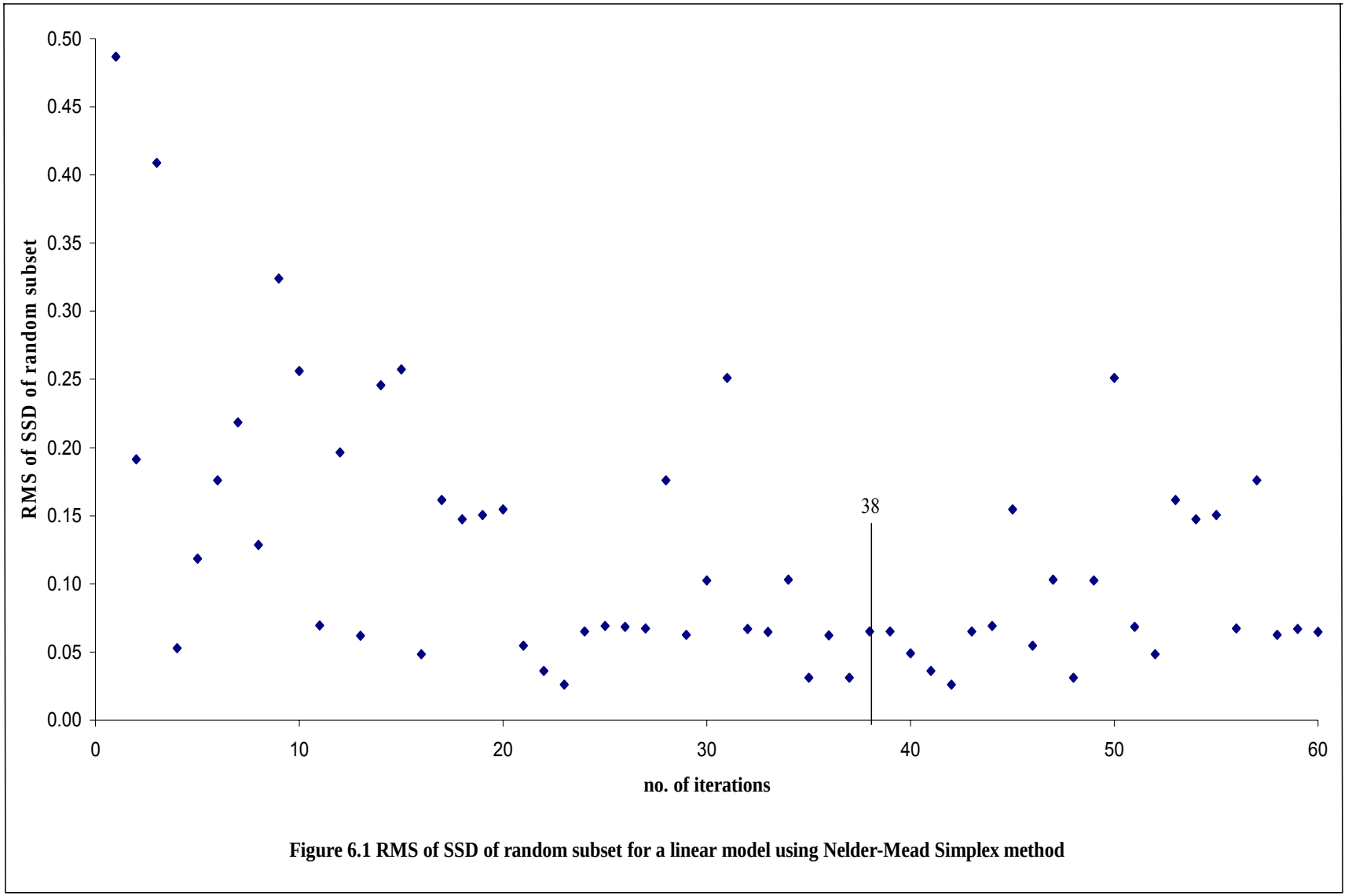


Figure 6.1 RMS of SSD of random subset for a linear model using Nelder-Mead Simplex method

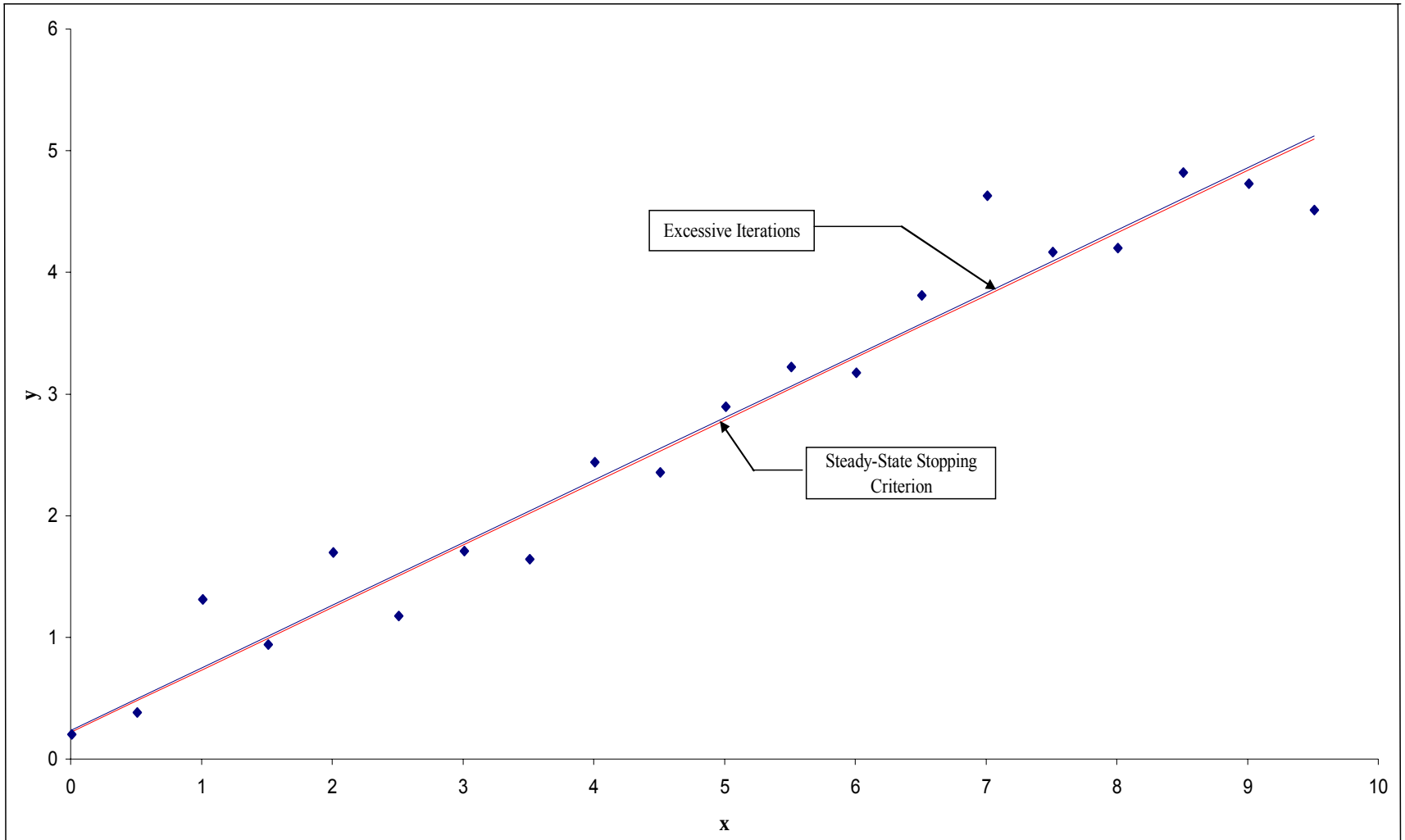


Figure 6.2 A comparison plot between the linear curves obtained from the two stopping criteria when using the Nelder-Mead Simplex method

Table 6.2: Parameter values for the linear model using Nelder-Mead Simplex method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	0.5	0.5142	0.5131
B	0.2	0.2328	0.2183

Case 6.1.1.2 Optimization Technique used: Marquardt-Levenberg method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Marquardt-Levenberg optimization method. The optimization procedure was run to obtain the optimum parametric values. Figure 6.3 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. Table 6.3 shows the F and p-statistic values for both the curves with respect to the originally generated noisy data. Both the curves and the generated noisy data are shown in Figure 6.4. From the visual evidence, it is clear that both the curves are indistinguishable.

Table 6.3: Goodness of fit for the linear model using Marquardt-Levenberg method

Test	Results
F-Statistic	0.9998
p-Value	0.4990

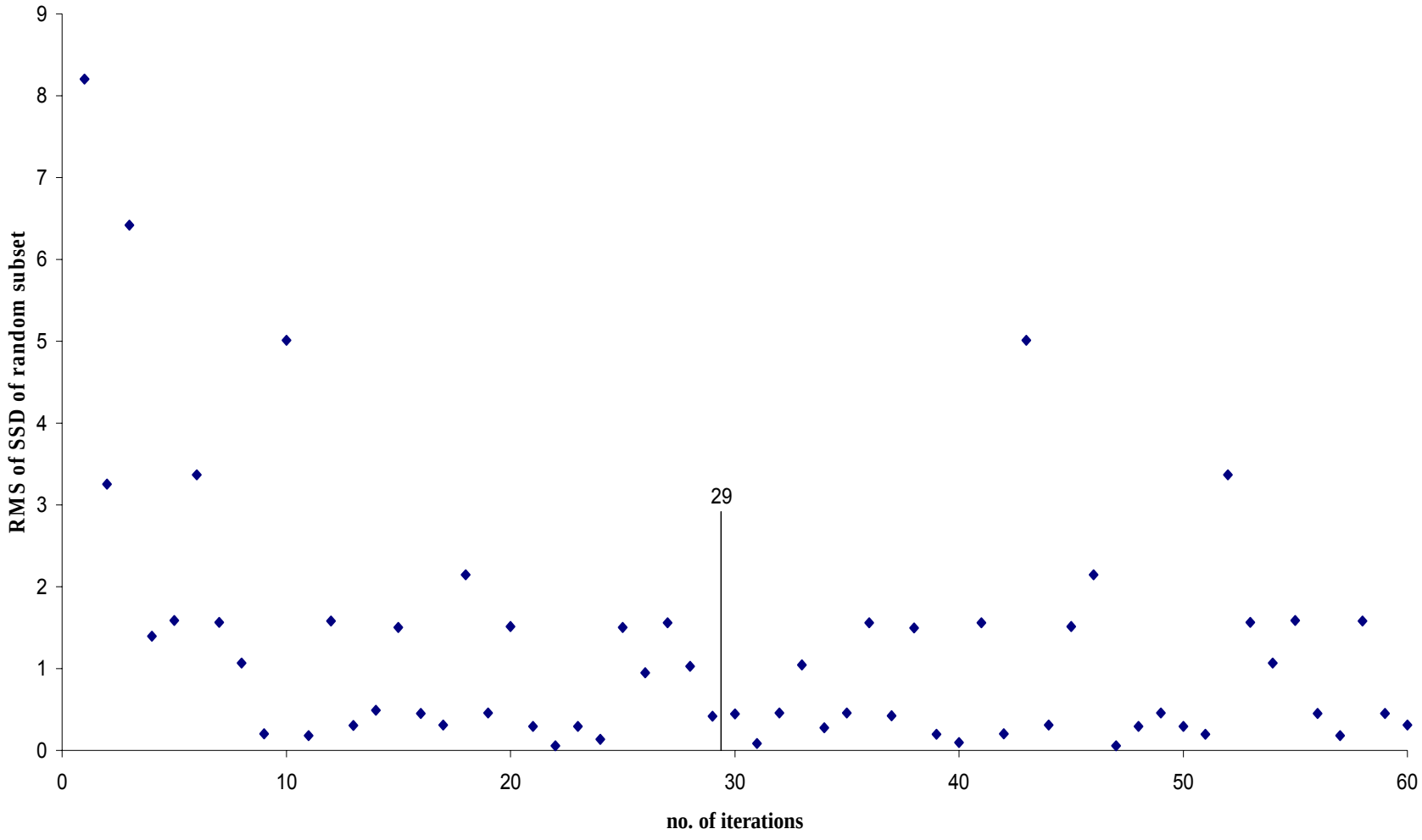


Figure 6.3 RMS of SSD of random subset for a linear model using Marquardt-Levenberg method

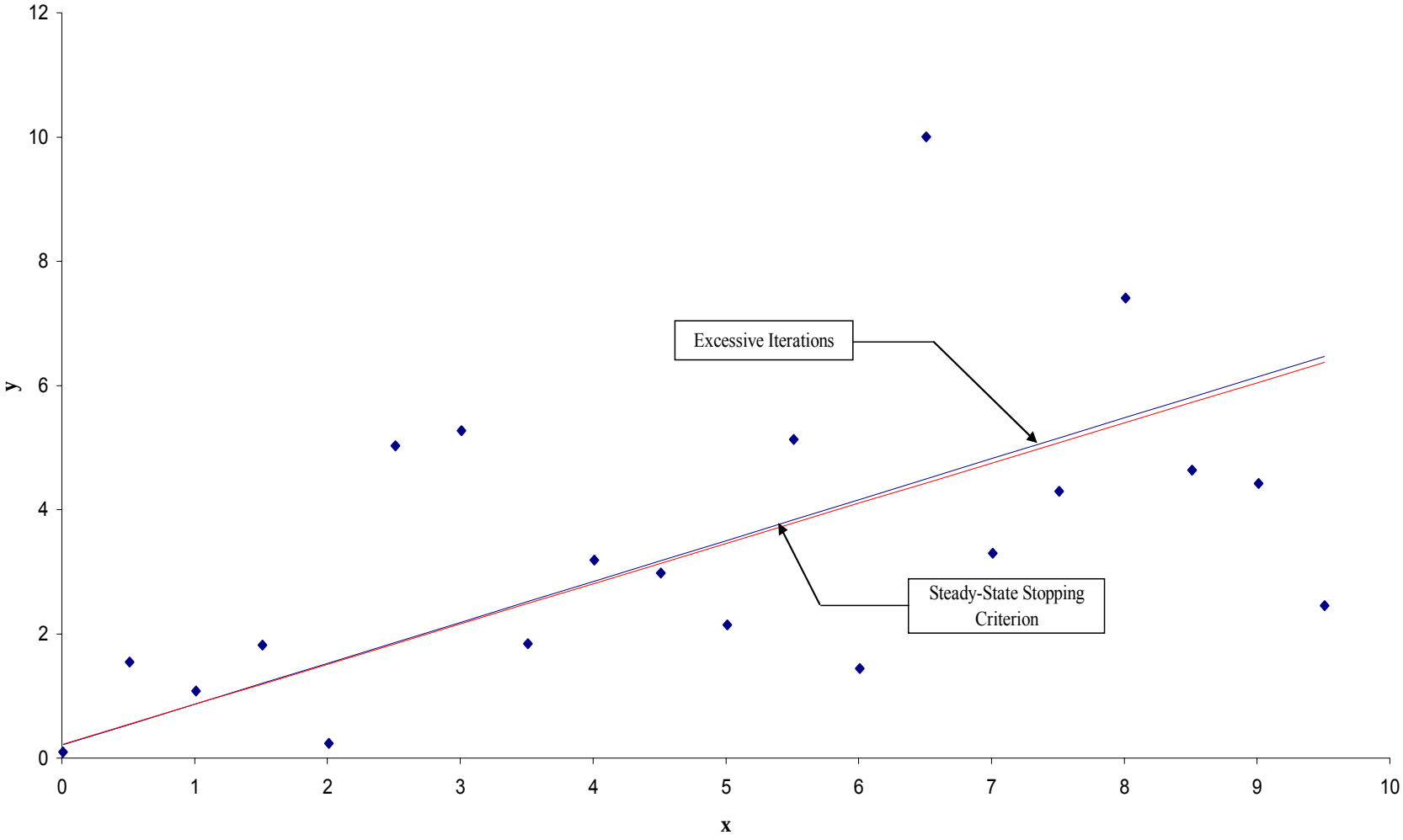


Figure 6.4 A comparison plot between the linear curves obtained from the two stopping criteria when using the Marquardt-Levenberg method

From Table 6.3, it can be concluded that both the curves obtained using the Marquardt-Levenberg method, are indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.4.

Table 6.4: Parameter values for the linear model using Marquardt-Levenberg method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	0.5	0.6587	0.6476
B	0.2	0.2073	0.2158

Case 6.1.1.3 Optimization Technique used: Gauss-Newton method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Gauss-Newton optimization method. The optimization procedure was run for an excessive number of iterations until no change in the SSD of the random subset was observed, to obtain the optimum parametric values. Figure 6.5 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. Table 6.3 shows the F and p-statistic values for both the curves with respect to the originally generated noisy data. Both the curves and the generated noisy data are shown in Figure 6.6. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

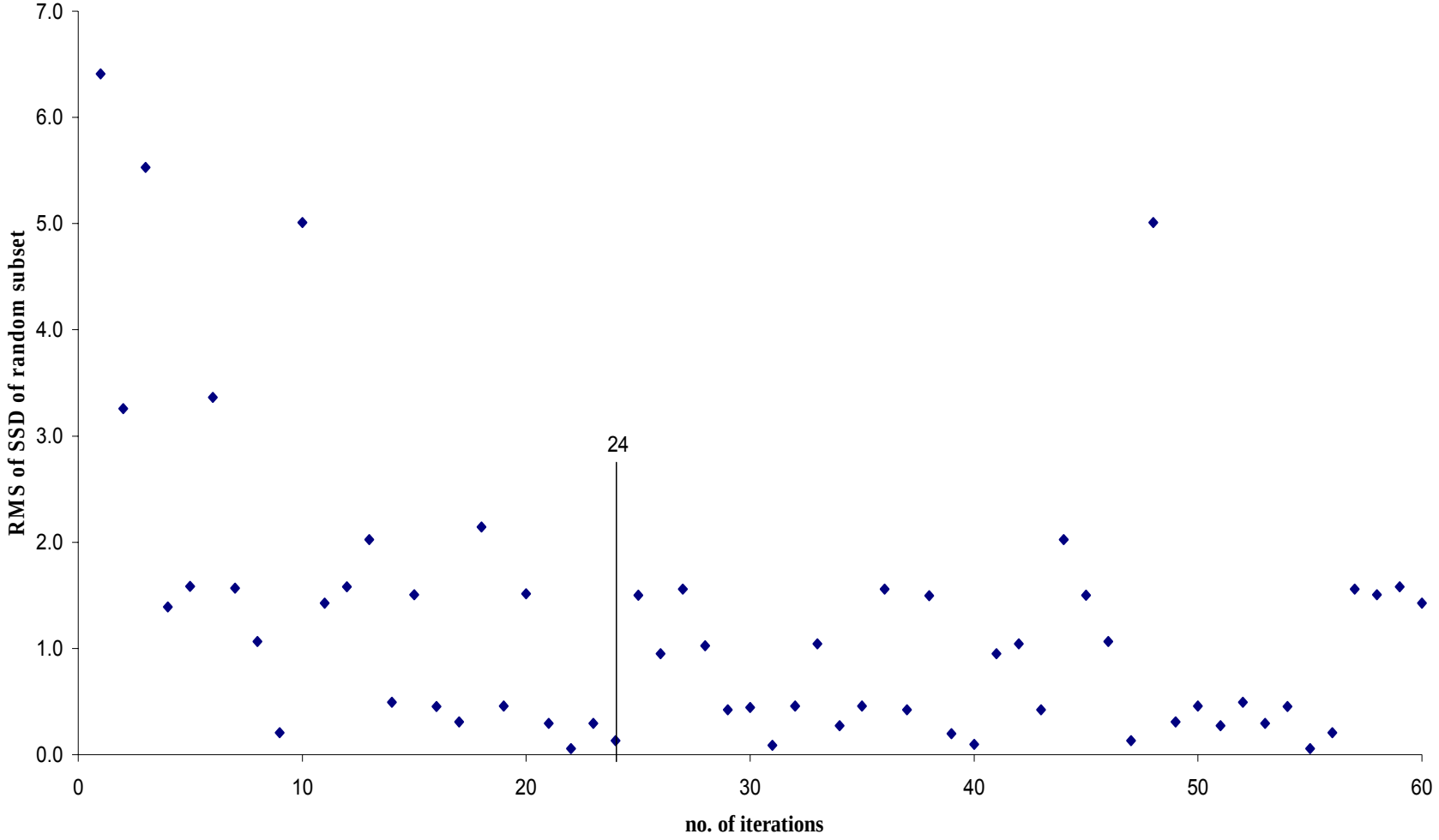


Figure 6.5 RMS of SSD of random subset for a linear model using Gauss-Newton method

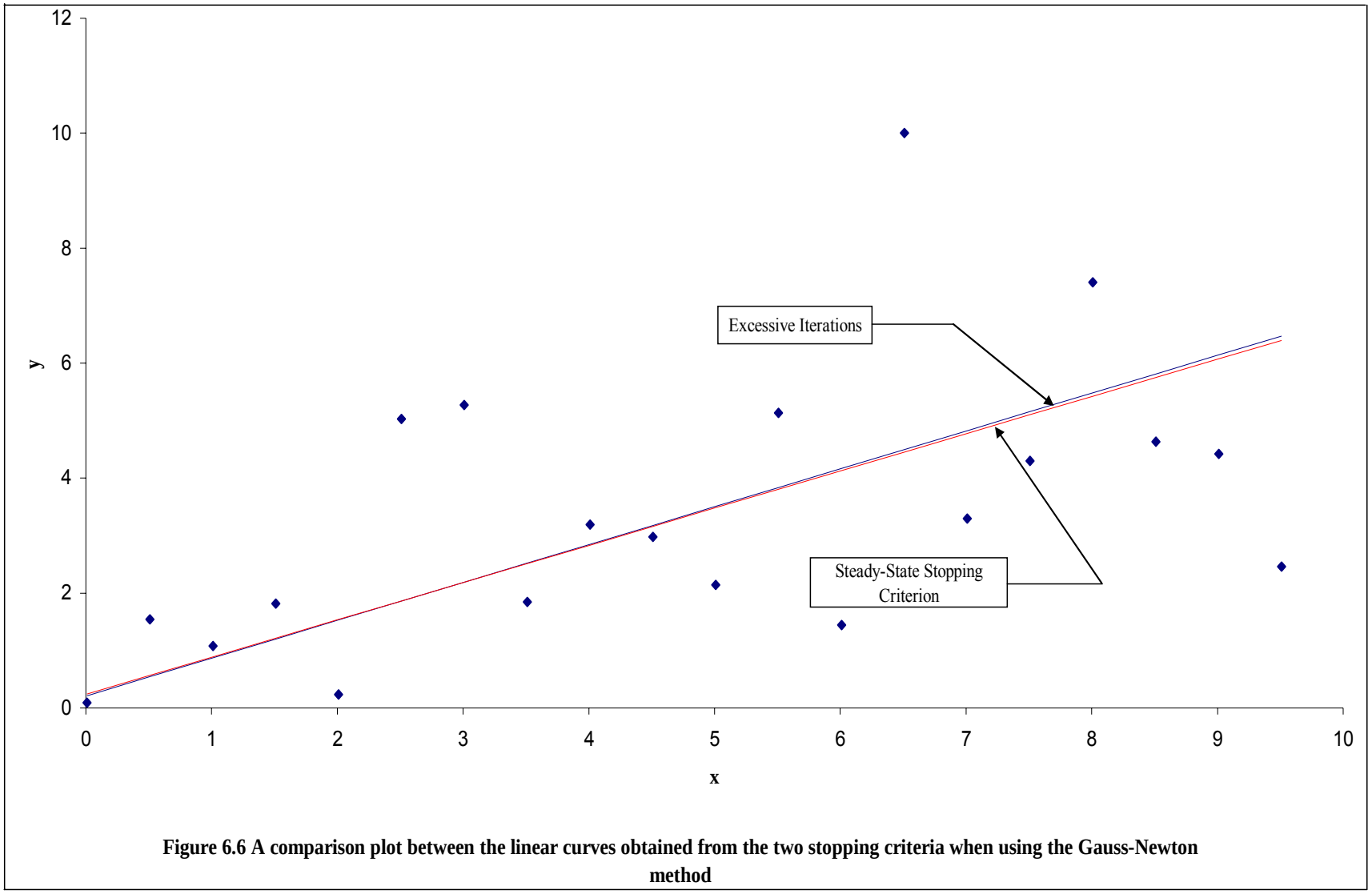


Figure 6.6 A comparison plot between the linear curves obtained from the two stopping criteria when using the Gauss-Newton method

Table 6.5: Goodness of fit for the linear model using Gauss-Newton method

Test	Results
F-Statistic	0.9997
p-Value	0.4990

From Table 6.5, it can be concluded that both the curves obtained using the Marquardt-Levenberg method, are indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.6.

Table 6.6: Parameter values for the linear model using Gauss-Newton method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	0.5	0.6587	0.6476
B	0.2	0.2073	0.2383

6.1.2 Optimization of Parameters in a Nonlinear Function

Nonlinear function used: $y = A\ln(Bx)$

Parameters to be optimized: A and B

The above mentioned nonlinear function was used to generate the data. The objective function values (y values) were calculated for A = 5 and B = 55 in a range of 'x' values

from 273 to 19773 with the interval of 500. Gaussian distributed random numbers [NID (0, 1)] were added to the above generated data using the random number generation code in MATLAB 6.5. The noisy data was then used by the optimizer to determine the best empirical values of A and B. The optimization code for different methods to optimize the parameters is written in MATLAB 6.5 release 13 (refer Appendix D). The optimization procedure was run for 75 iterations and the parameter values obtained were recorded to calculate the objective function values. The excessive number of iterations was decided on the basis of change in the sum of squared deviations of the random subset. Another set of parameter values was obtained at a point where the novel stopping criterion suggested termination. The results obtained using the three optimization techniques, *viz*. Nelder-Mead Simplex method, Marquardt-Levenberg method and the Gauss-Newton method are discussed in cases below.

Case 6.1.2.1 Optimization Technique used: Nelder-Mead Simplex

Three random initial values, to form the first simplex, were given to each of the parameters that are to be optimized using the Nelder-Mead Simplex method. The optimization procedure was then run for an excessive number of iterations until no change in the SSD of the random subset was observed. The plot showing the change in the sum of squared deviations of the random subset with the iterations is shown in Figure 6.7. The number of iterations, took to obtain the optimum values of the parameters using the novel stopping criterion, is clearly indicated in Figure 6.7. The objective function values that resulted from the latter set of parameter values were compared to that obtained from the former using the F and p-statistics. The F and p-statistics and the

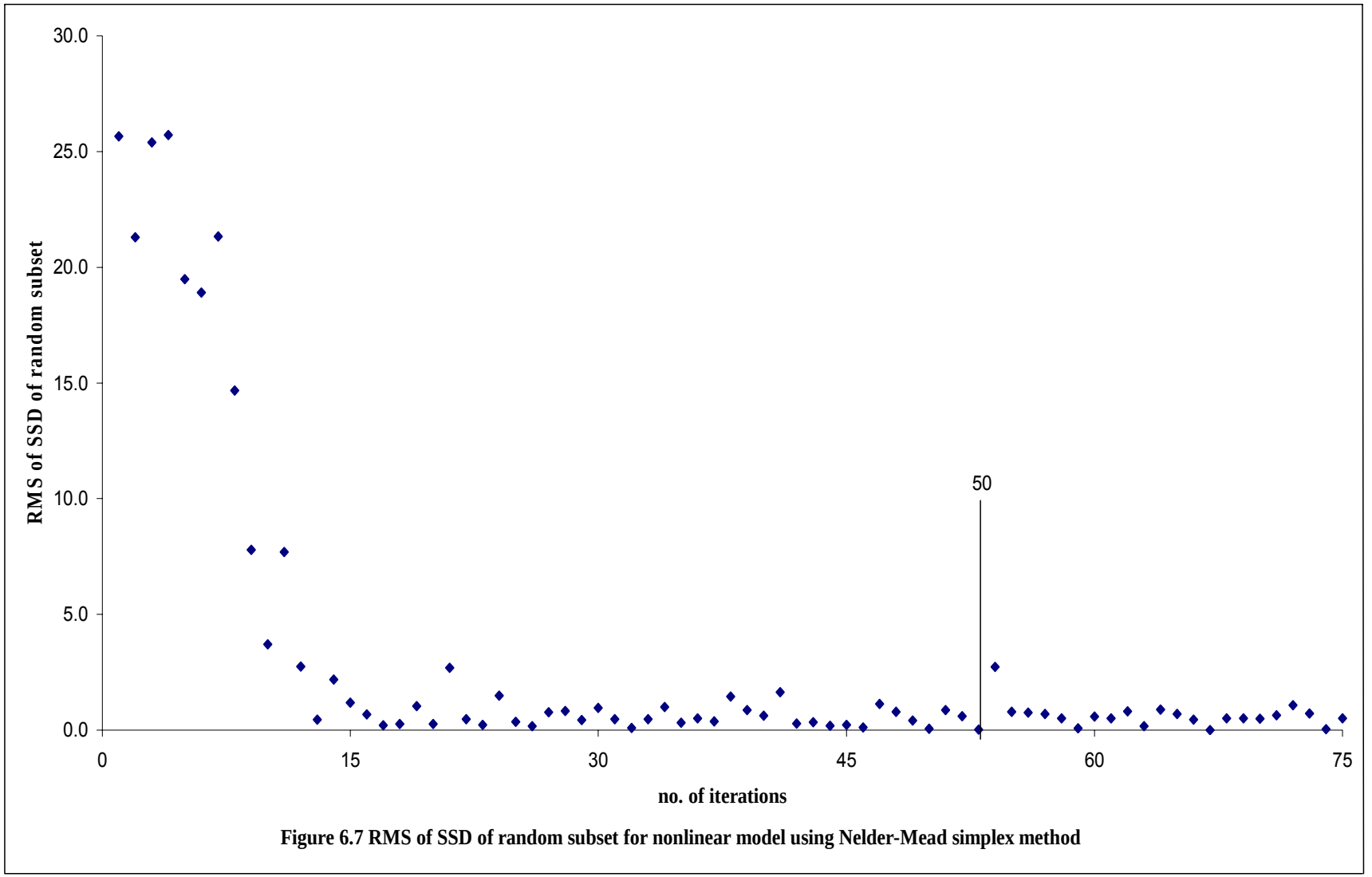


Figure 6.7 RMS of SSD of random subset for nonlinear model using Nelder-Mead simplex method

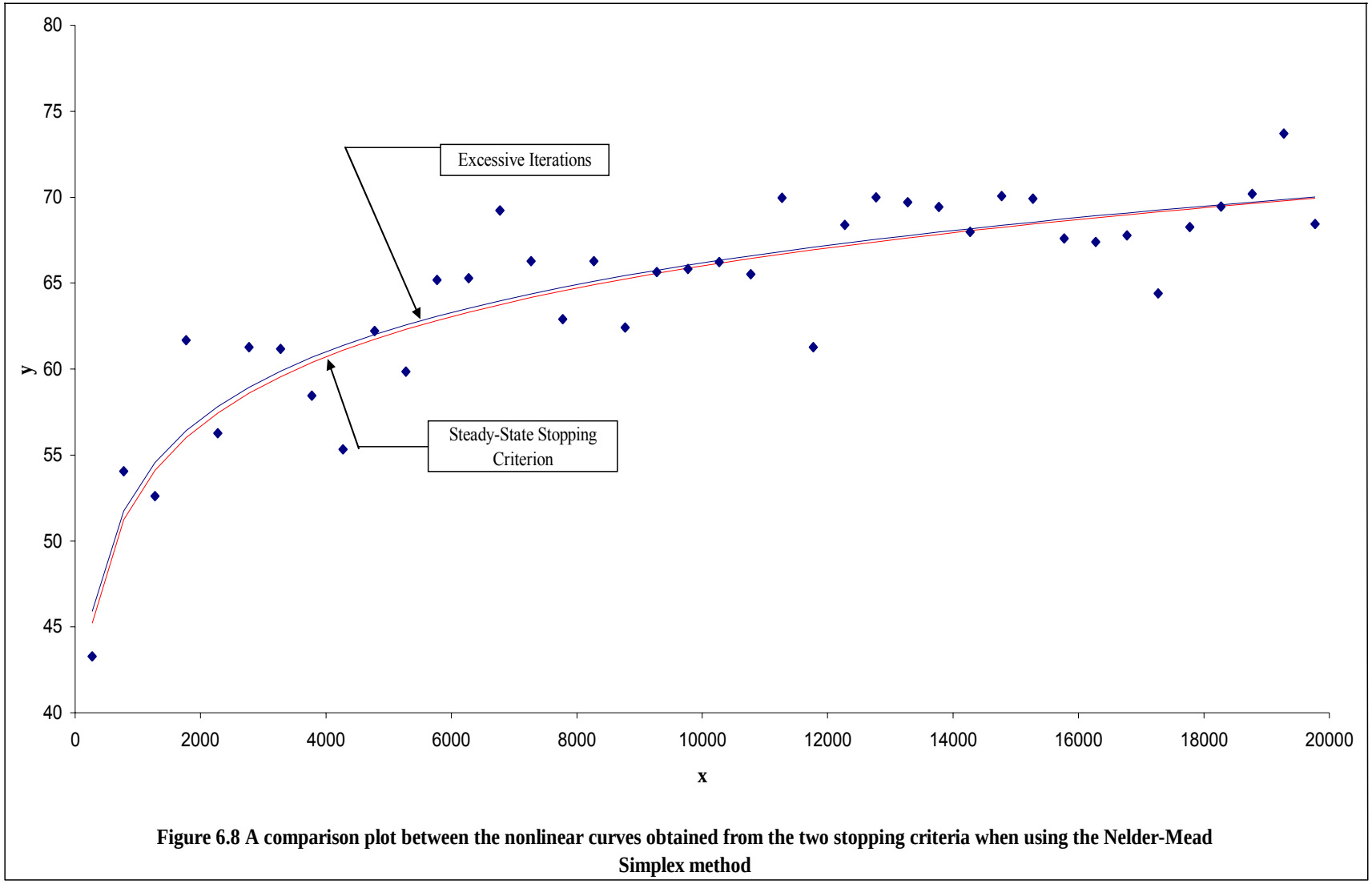


Figure 6.8 A comparison plot between the nonlinear curves obtained from the two stopping criteria when using the Nelder-Mead Simplex method

parameter values for both the curves are shown in Tables 6.7 and 6.8. The comparison plot is shown in Figure 6.8. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

Table 6.7: Goodness of fit for the nonlinear model using Nelder-Mead Simplex method

Test	**Results**
F-Statistic	0.9907
p-Value	0.4990

From Table 6.7, it can be observed that the F-statistic is close to unity which in turn suggests that both the curves are statistically indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.8.

Table 6.8: Parameter values for the nonlinear model using Nelder-Mead Simplex method

Parameters	**Model Values**	**Excessive Iterations**	**Steady-State stopping Criterion**
A	5	5.6326	5.7692
B	55	12.6358	9.2858

Case 6.1.2.2 Optimization Technique used: Marquardt-Levenberg method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Marquardt-Levenberg optimization method. The optimization procedure was run to obtain the optimum parametric values. Figure 6.9 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. Table 6.9 shows the F and p-statistic values for both the curves with respect to the originally generated noisy data. Both the curves and the generated noisy data are shown in Figure 6.10. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

Table 6.9: Goodness of fit for the nonlinear model using Marquardt-Levenberg method

Test	**Results**
F-Statistic	0.9998
p-Value	0.4990

From Table 6.9, it can be concluded that both the curves obtained using the Marquardt-Levenberg method, are indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.10.

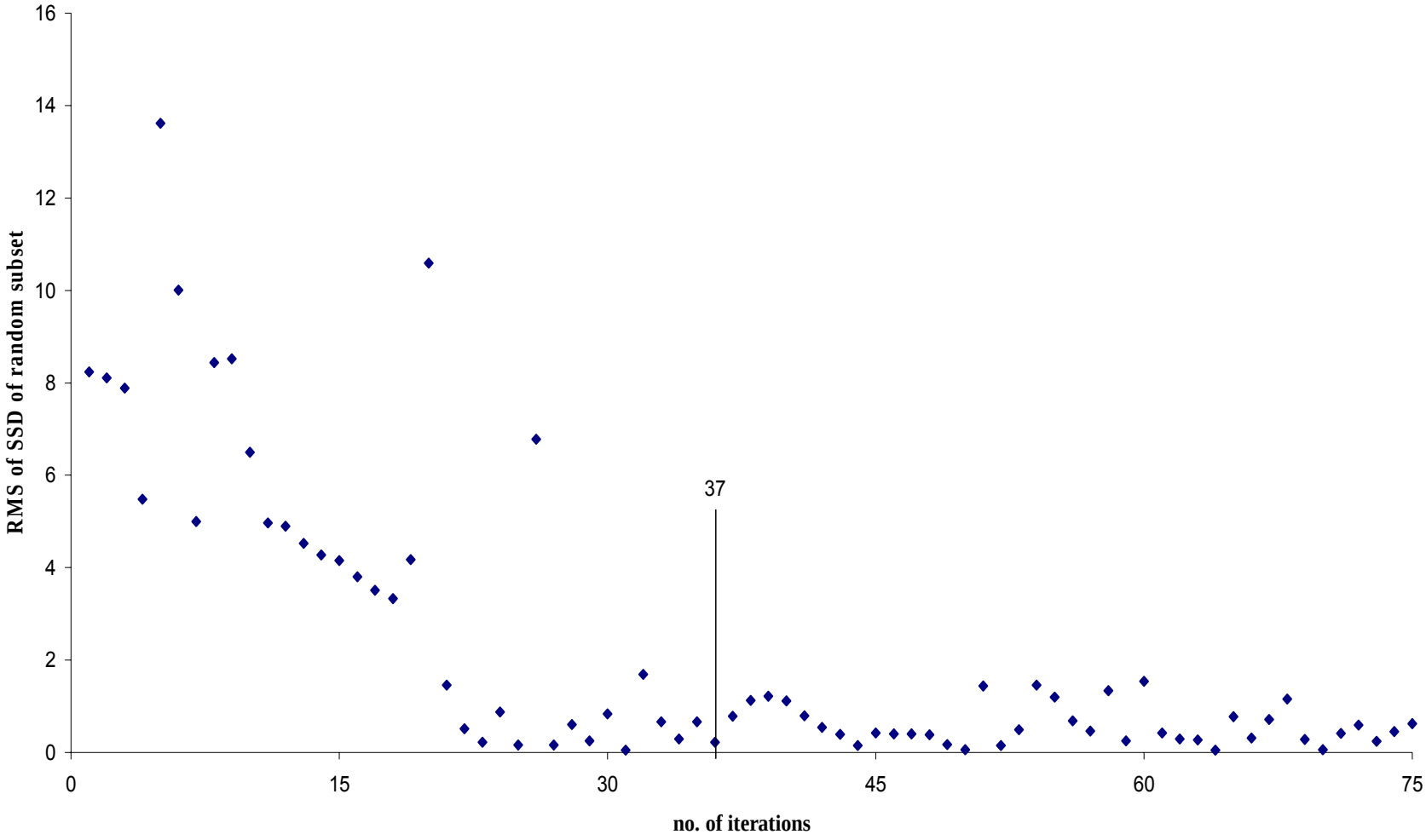


Figure 6.9 RMS of SSD of random subset for nonlinear model using Marquardt-Levenberg method

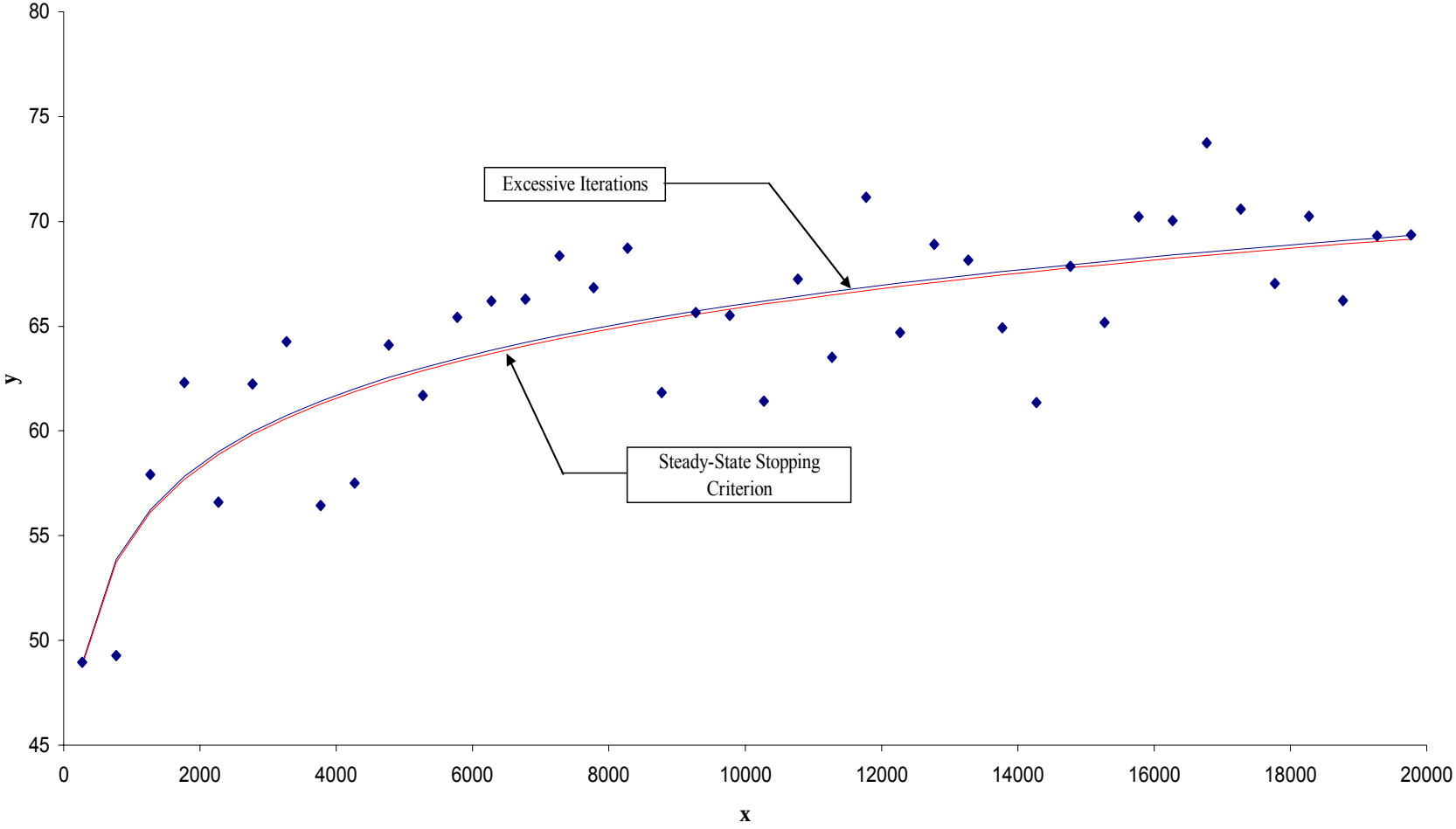


Figure 6.10 A comparison plot between the nonlinear curves obtained from the two stopping criteria when using the Marquardt-Levenberg method

Table 6.10: Parameter values for the nonlinear model using Marquardt-Levenberg method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	5	4.7678	4.7566
B	55	104.3630	104.3840

Case 6.1.2.3 Optimization Technique used: Gauss-Newton method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Gauss-Newton optimization method. The optimization procedure was run for an excessive number of iterations until no change in the SSD of the random subset was observed, to obtain the optimum parametric values. Figure 6.11 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. Table 6.11 shows the F and p-statistic values for both the curves with respect to the originally generated noisy data. Both the curves and the generated noisy data are shown in Figure 6.12. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

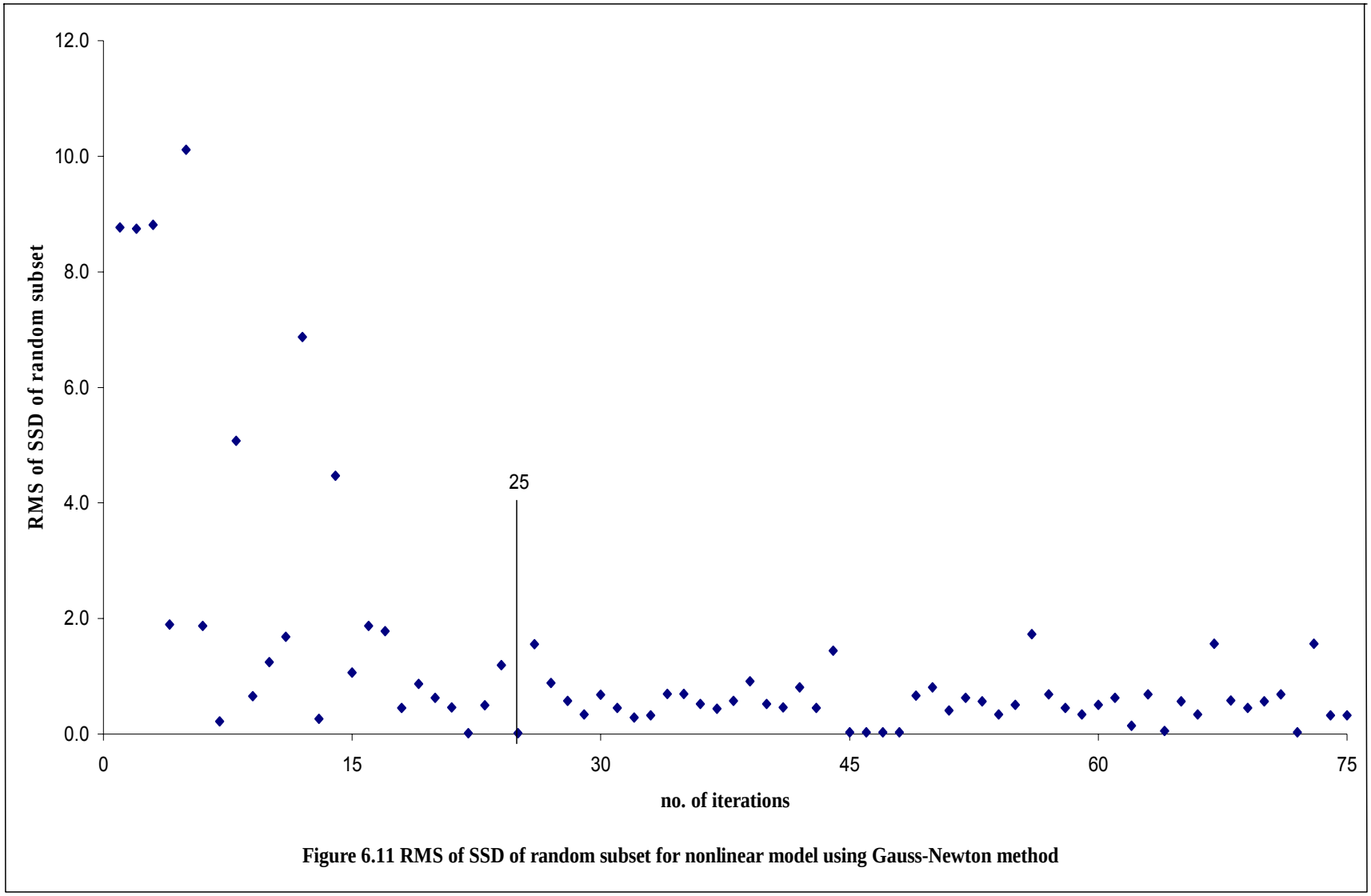


Figure 6.11 RMS of SSD of random subset for nonlinear model using Gauss-Newton method

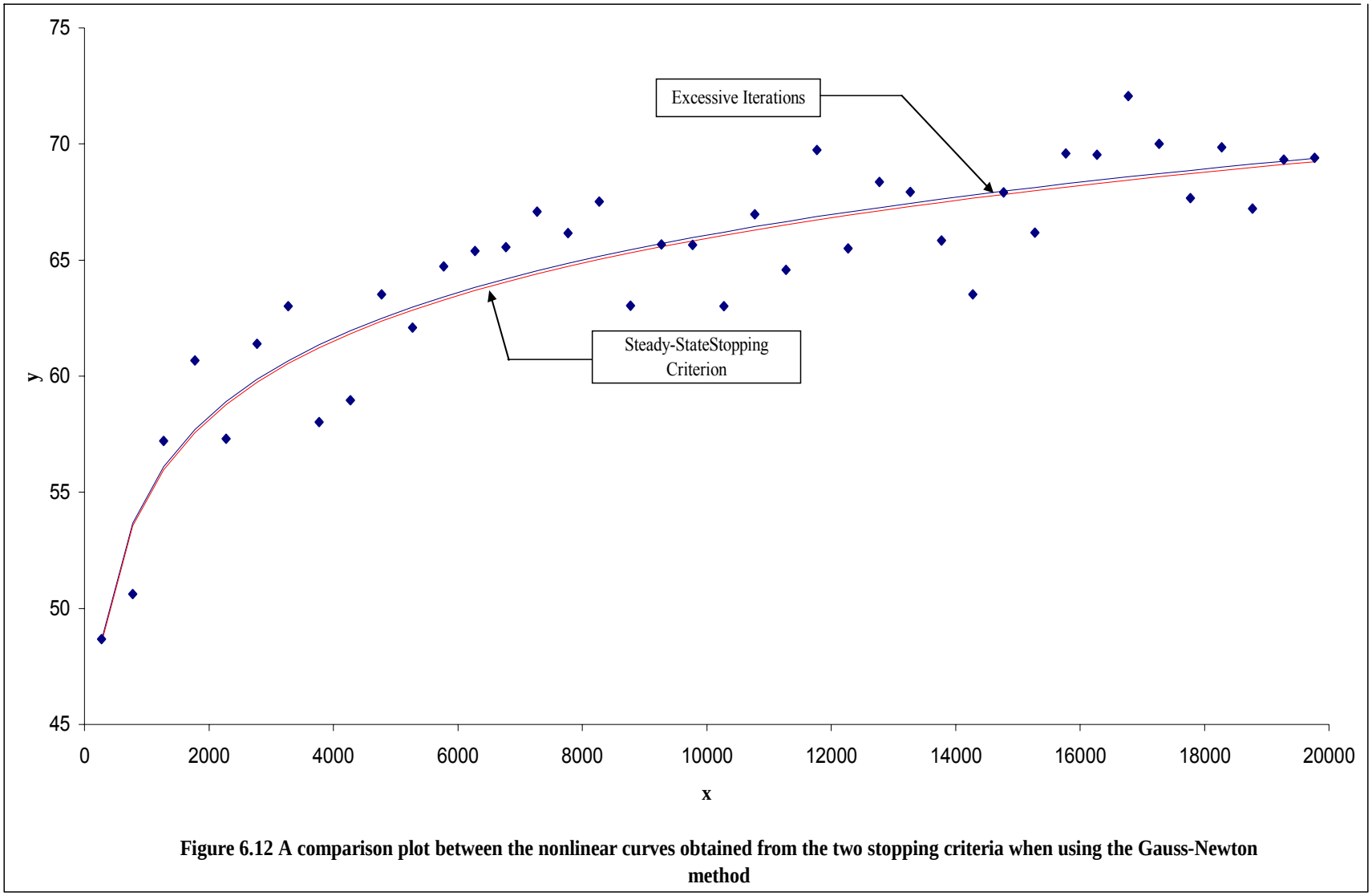


Figure 6.12 A comparison plot between the nonlinear curves obtained from the two stopping criteria when using the Gauss-Newton method

Table 6.11: Goodness of fit for the nonlinear model using Gauss-Newton method

Test	Results
F-Statistic	0.9997
p-Value	0.4990

From Table 6.11, it can be concluded that both the curves obtained using the Marquardt-Levenberg method, are indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.12.

Table 6.12: Parameter values for the linear model using Gauss-Newton method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	5	4.8452	4.8352
B	55	83.7278	83.7368

6.1.3 Optimization of Parameters in a multivariable nonlinear Function

Nonlinear function used: $z = A\sqrt{x} + B\sqrt{y}$

Parameters to be optimized: A and B

The above mentioned nonlinear function was used to generate the data. The objective function values (*y* values) were calculated for A = 0.5 and B = 2 in a range of '*x*' values

from 0 to 10 with the interval of 0.5. Gaussian distributed random numbers [NID (0,1)] were added to the above generated data using the random number generation code in MATLAB 6.5. The noisy data was then used by the optimizer to determine the best empirical values of A and B. The optimization code for different methods to optimize the parameters is written in MATLAB 6.5 release 13 (refer Appendix D). The optimization procedure was run for 60 iterations and the parameter values obtained were recorded to calculate the objective function values. The excessive number of iterations was decided on the basis of change in the sum of squared deviations of the random subset. Another set of parameter values was obtained at a point where the novel stopping criterion suggested termination. The results obtained using the three optimization techniques, *viz*. Nelder-Mead Simplex method, Marquardt-Levenberg method and the Gauss-Newton method are discussed in cases below.

Case 6.1.3.1 Optimization Technique used: Nelder-Mead Simplex

Three random initial values, to form the first simplex, were given to each of the parameters that are to be optimized using the Nelder-Mead Simplex method. The optimization procedure was then run for an excessive number of iterations until no change in the SSD of the random subset was observed. The plot showing the change in the sum of squared deviations of the random subset with the iterations is shown in Figure 6.13. The number of iterations, took to obtain the optimum values of the parameters using the novel stopping criterion, is clearly indicated in Figure 6.13. The objective function values that resulted from the latter set of parameter values were compared to that obtained from the former using the F and p-statistics. The F and p-statistics and the

parameter values for both the curves are shown in Tables 6.13 and 6.14. The comparison plot is shown in Figure 6.14. The black and the white markers indicate that the points are above and below the plane, respectively. The dark shading on the surface show that the two surfaces overlap. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

Table 6.13: Goodness of fit for the multivariable nonlinear model using Nelder-Mead Simplex method

Test	Results
F-Statistic	0.9146
p-Value	0.4190

From Table 6.13, it can be observed that the F-statistic is close to unity and the p value is close to 0.5 which suggests that both the curves are statistically indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.14.

Table 6.14: Parameter values for the multivariable nonlinear model using Nelder-Mead Simplex method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	0.5	0.1582	0.1652
B	2	2.3414	2.3916

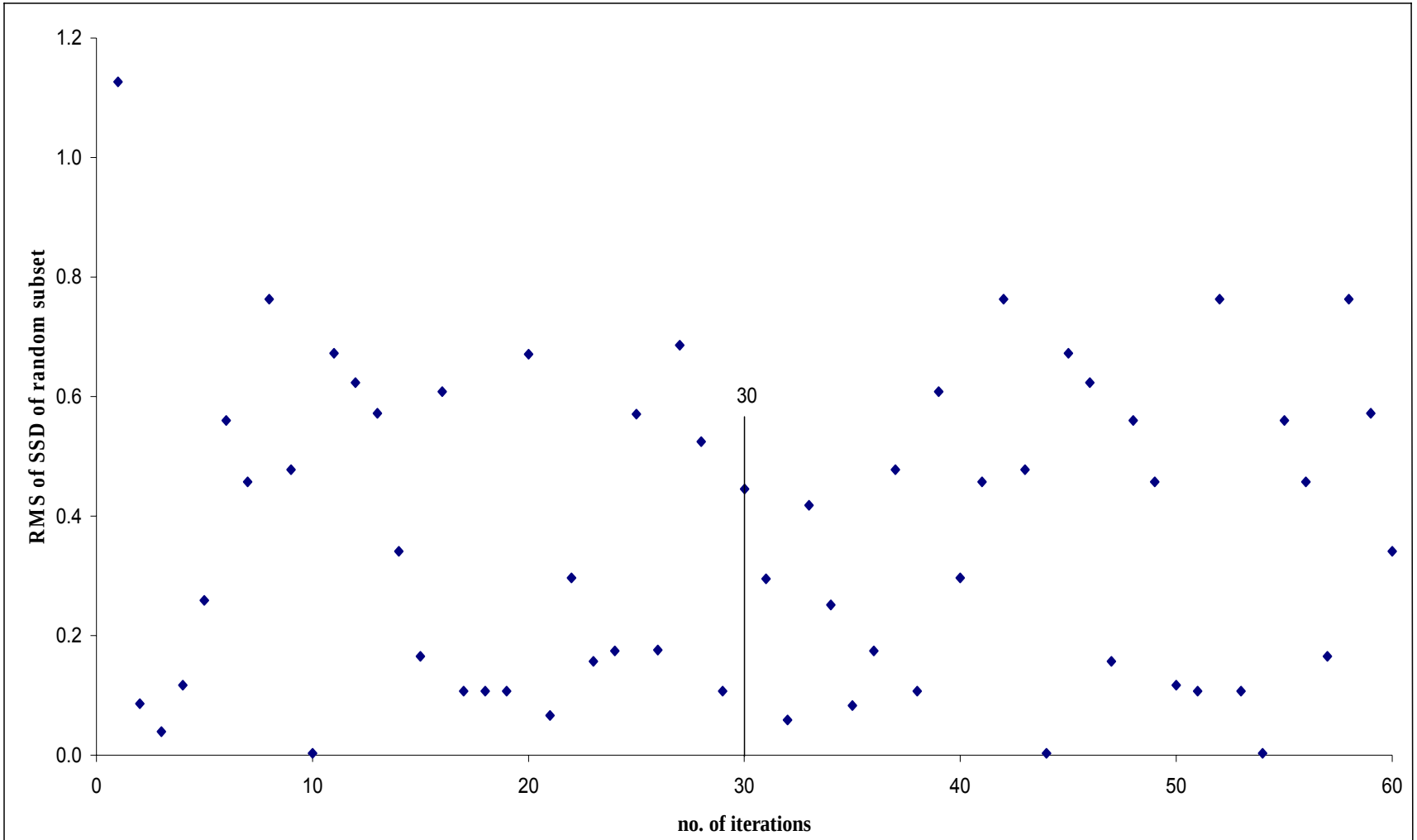


Figure 6.13 RMS of SSD of random subset for multivariable model using Nelder-Mead Simplex method

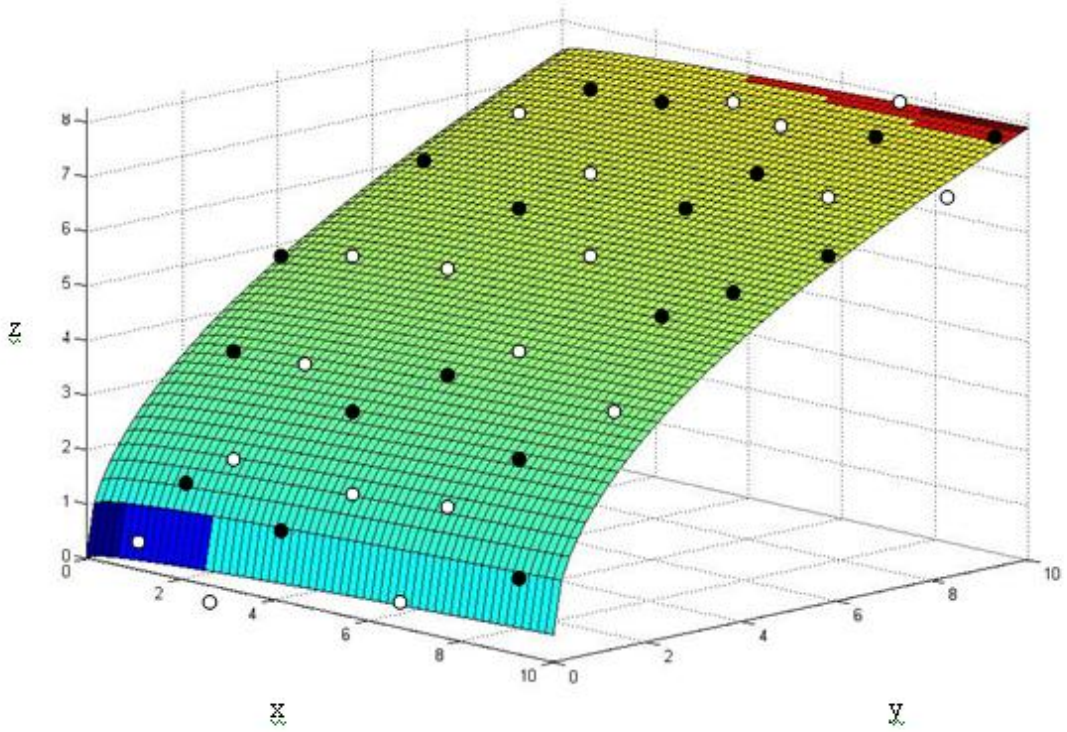


Figure 6.14 A comparison plot between the multivariable curves obtained from the two stopping criteria when using the Nelder-Mead simplex method

Case 6.1.3.2 Optimization Technique used: Marquardt-Levenberg method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Marquardt-Levenberg optimization method. The optimization procedure was run to obtain the optimum parametric values. Figure 6.15 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. Table 6.15 shows the F and p-statistic values for both the curves with respect to the originally generated noisy data. Both the curves and the generated noisy data are shown in Figure 6.16. The black and the white markers indicate that the points are above and below the plane respectively. The dark shading on the surface show that the two surfaces overlap. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

Table 6.15: Goodness of fit for the multivariable nonlinear model using Marquardt-Levenberg method

Test	**Results**
F-Statistic	0.9983
p-Value	0.4980

From Table 6.15, it can be concluded that both the curves obtained using the Marquardt-Levenberg method, are indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.16.

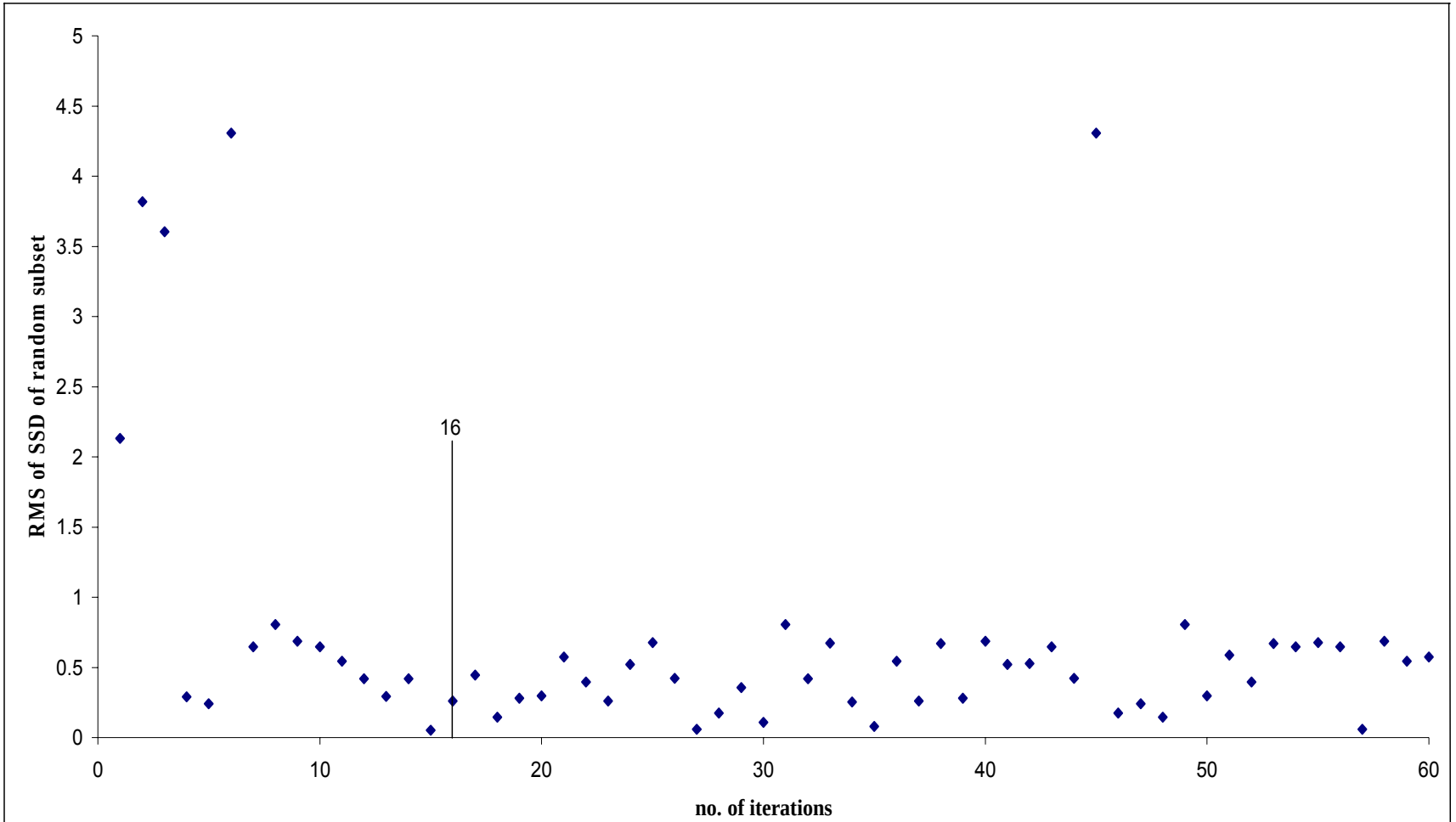


Figure 6.15 RMS of SSD of random subset for multivariable model using Marquardt-Levenberg method

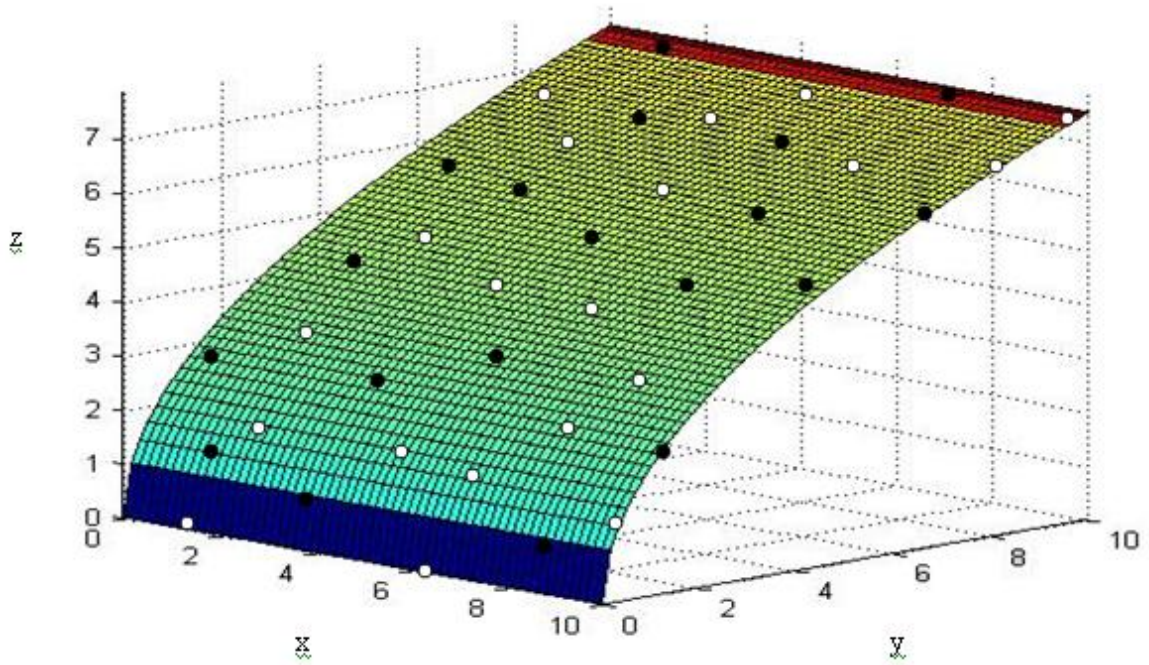


Figure 6.16 A comparison plot between the multivariable curves obtained from the two stopping criteria when using the Marquardt-Levenberg method

Table 6.16: Parameter values for the multivariable nonlinear model using Marquardt-Levenberg method

Parameters	Model Values	Excessive Iterations	Steady-State stopping Criterion
A	0.5	1.3992	1.3562
B	2	1.1005	1.0225

Case 6.1.3.3 Optimization Technique used: Gauss-Newton method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Gauss-Newton optimization method. The optimization procedure was run for an excessive number of iterations until no change in the SSD of the random subset was observed, to obtain the optimum parametric values. Figure 6.17 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. Table 6.17 shows the F and p-statistic values for both the curves with respect to the originally generated noisy data. Both the curves and the generated noisy data are shown in Figure 6.18. The black and the white markers indicate that the points are above and below the plane respectively. The dark shading on the surface show that the two surfaces overlap. From the visual evidence, it is clear that both the curves are indistinguishable relative to variance in the data.

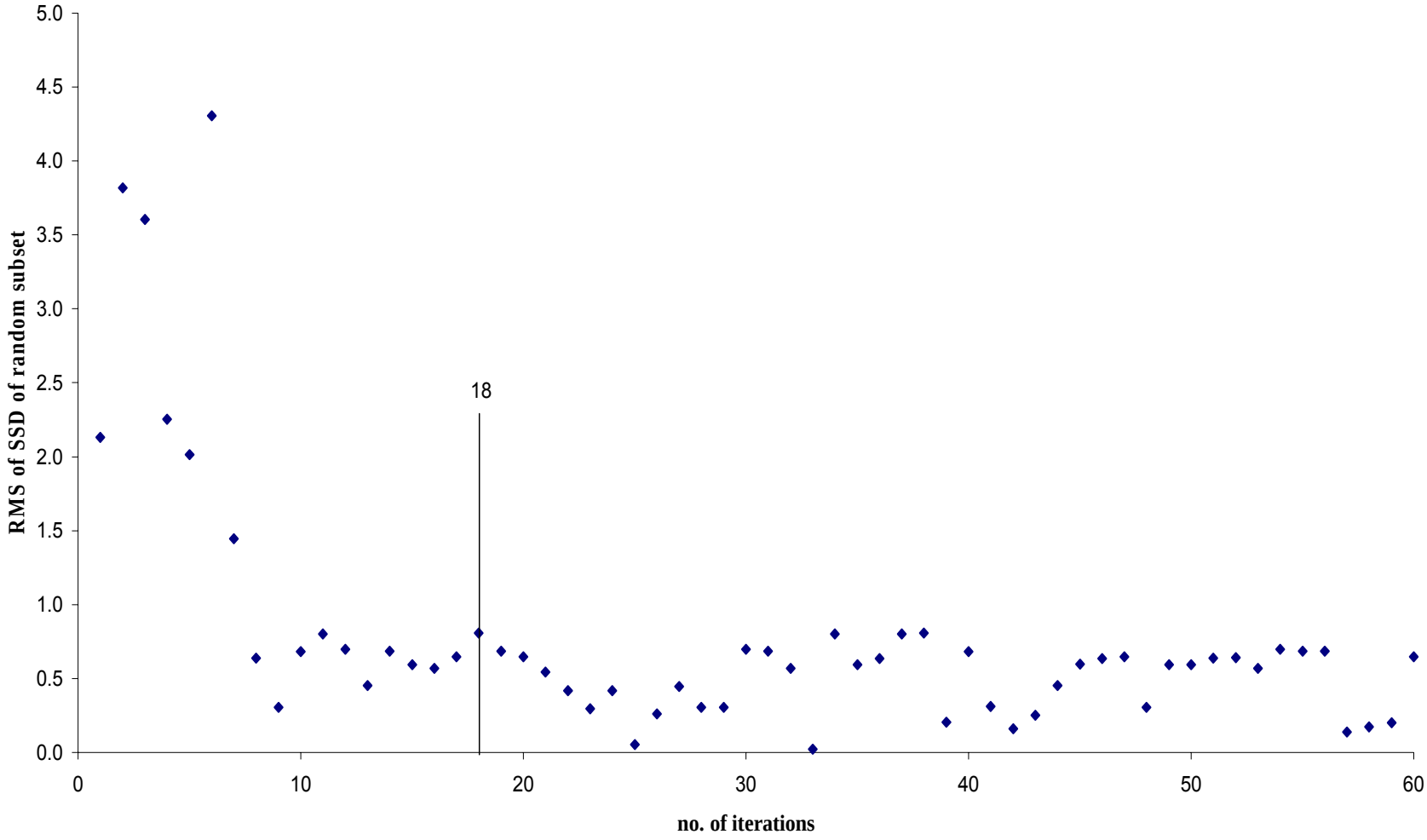


Figure 6.17 RMS of SSD of random subset for multivariable model using Gauss-Newton method

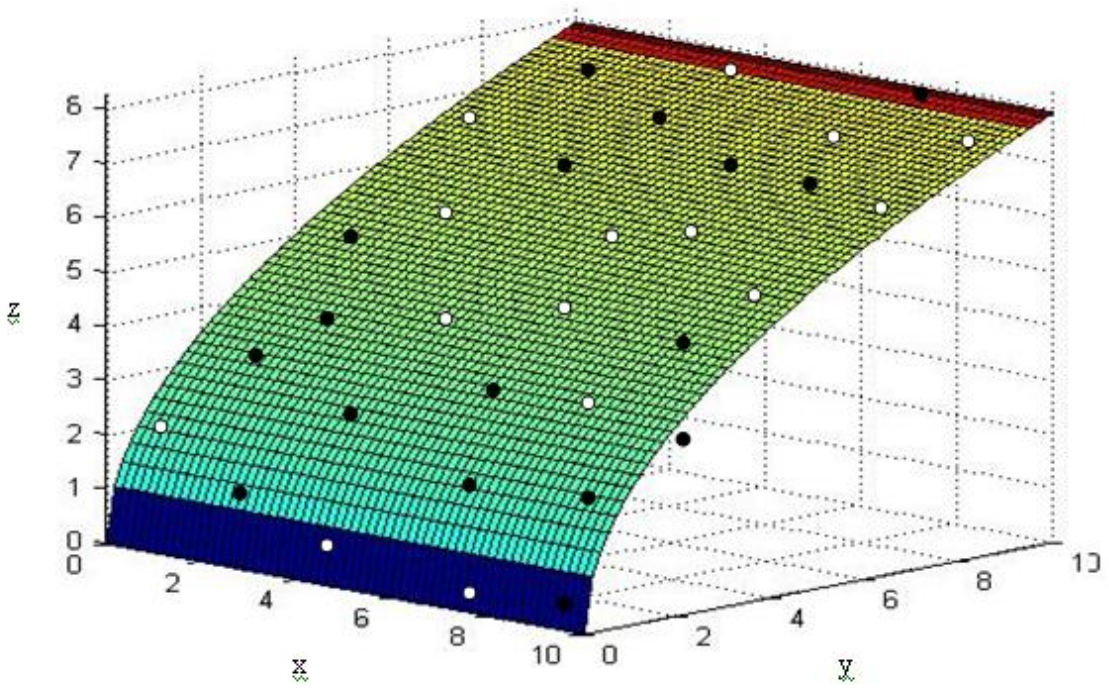


Figure 6.18 A comparison plot between the multivariable curves obtained from the two stopping criteria when using the Gauss-Newton method

Table 6.17: Goodness of fit for the multivariable nonlinear model using Gauss-Newton method

Test	Results
F-Statistic	0.9661
p-Value	0.4680

From Table 6.17, it can be concluded that both the curves obtained using the Marquardt-Levenberg method, are indistinguishable. The parametric values obtained for both the curves are listed below in Table 6.18.

Table 6.18: Parameter values for the multivariable nonlinear model using Gauss-Newton method

Parameters	Model Values	Excessive Iterations	SS stopping Criterion
A	0.5	1.3992	1.3982
B	2	1.1105	1.102

The optimization procedure was run with different seed values to the random number generator which was used to add noise to the data. It was observed that the change in the seed values to generate random numbers to add noise to the data did not have a great effect on the optimization. Table 6.19 shows the parameter values obtained from different seed values using the Gauss-Newton method for the multivariable model.

Table 6.19: Parameter values for the multivariable nonlinear model using Gauss-Newton method using different seed values

Seed Values	Seed = 0		Seed = 1		Seed = 2	
	A	B	A	B	A	B
SS Criterion	1.3982	1.102	1.4265	1.1365	1.4132	1.1956

6.2 Results from the Experimental Data

The experimental data was obtained by carrying out two laboratory scale experiments-the decomposition of methyl acetate in packed bed reactor (PBR) and the vapor-liquid two-phase flow experiment.

6.2.1 Optimization of Parameters in the Rate Equation

A decomposition reaction, where in methyl acetate is decomposed to give methanol and acetic acid in a packed bed reactor (PBR), is used to obtain the data required to calculate the output concentration of methyl acetate.

Case 6.2.1.1 Optimization of parameters in the reaction kinetic model using Nelder-Mead Simplex method

Three random initial values, to form the first simplex, were given to each of the parameters that are to be optimized using the Nelder-Mead Simplex method. The optimization procedure was then run for an excessive number of iterations of about 400, until no change in the SSD of the random subset was observed. The plot showing the change in the root mean square of the sum of squared deviations of the random subset (RMS SSD RS) with the iterations is shown in Figure 6.19. The number of iterations, took to obtain the optimum values of the parameters using the novel stopping criterion, is clearly indicated in Figure 6.19. The objective function values that resulted from the latter set of parameter values were compared to that obtained from the former using the mean sum of squared distances. The mean sum of squares distances of the data points from the 'x = y' line is shown in Table 6.19. The comparison plot is shown in Figure 6.20.

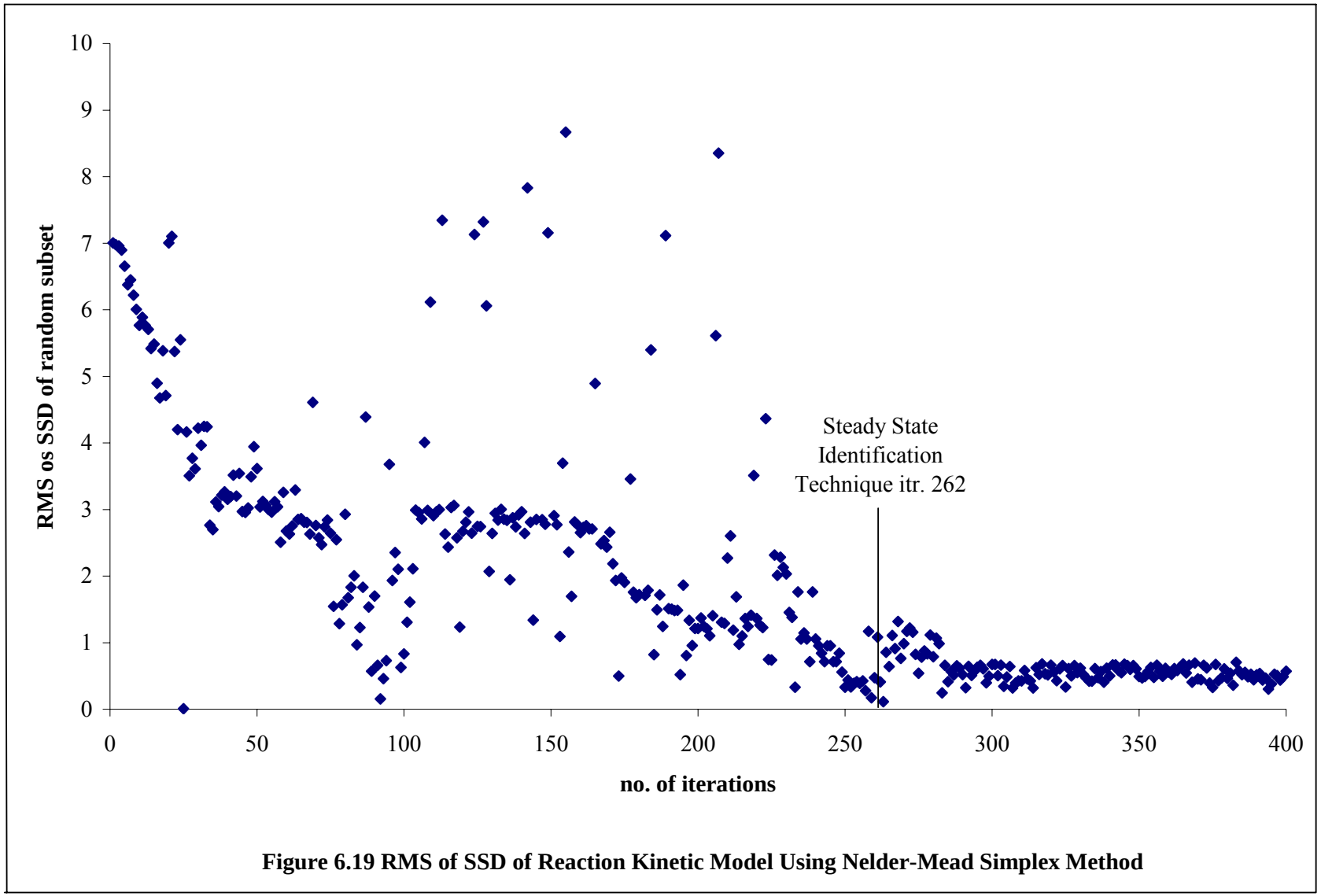


Figure 6.19 RMS of SSD of Reaction Kinetic Model Using Nelder-Mead Simplex Method

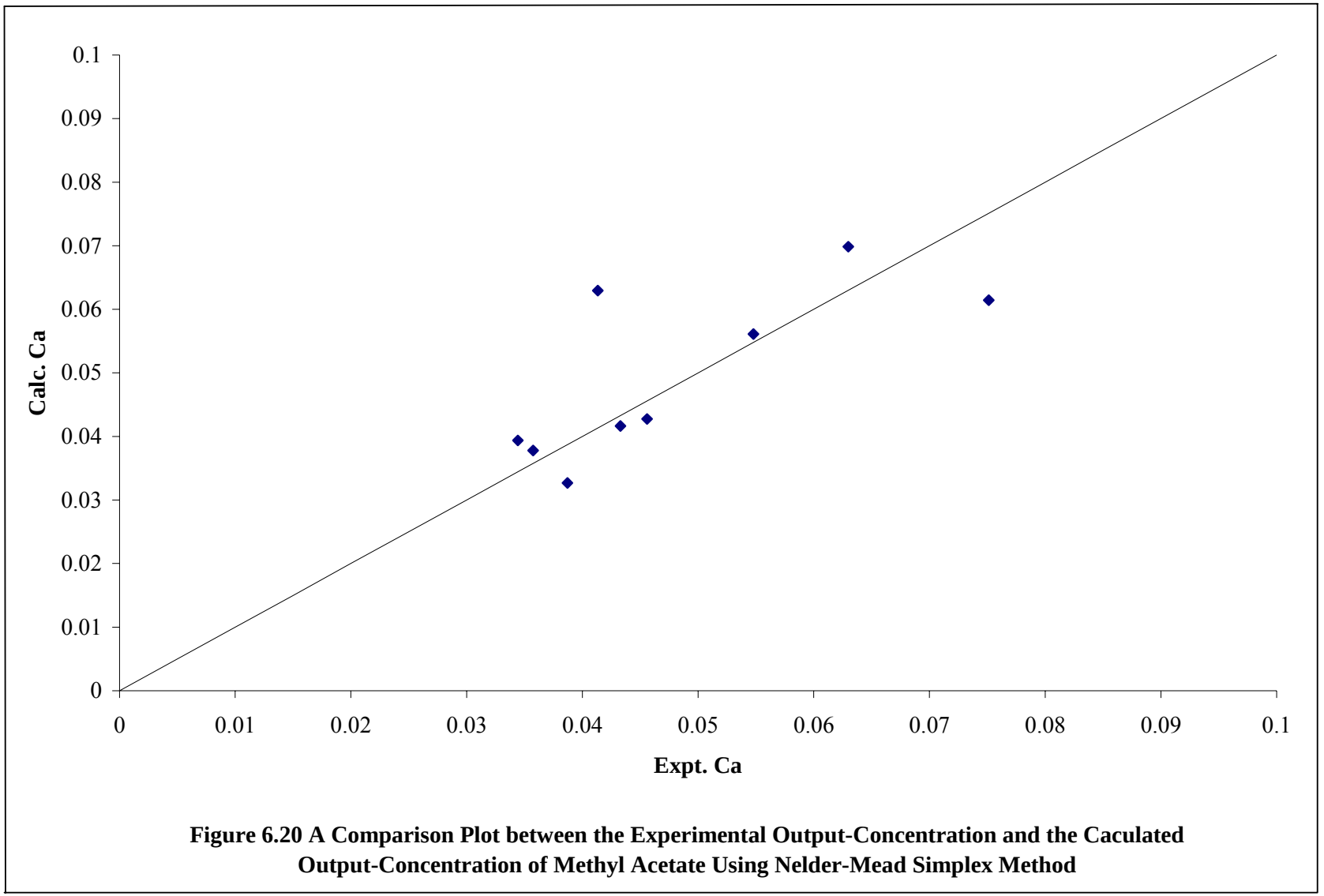


Figure 6.20 A Comparison Plot between the Experimental Output-Concentration and the Caculated Output-Concentration of Methyl Acetate Using Nelder-Mead Simplex Method

Table 6.20: Mean Sum of Squares Distances of Data Points from 'x = y' Line using Nelder-Mead Simplex Method

Stopping Criteria	Mean Sum of Squared Distances
Excessive Iterations	0.00680
SS Technique	0.00762

From Table 6.19, it can be observed that the mean sum of squared distances is almost equal for both criteria. The parametric values obtained for both the curves are listed below in Table 6.20.

Table 6.21: Parameter values for the reaction kinetic model using Nelder-Mead Simplex method

Parameters	Excessive Iterations	Steady-State stopping Criterion
A (1/s)	9.4253 E+6	9.4266 E+6
E (J/mol)	5947.215	5897.406
k_b (1/s)	2.4320	2.2465

Case 6.2.1.2 Optimization of parameters in the reaction kinetic model using Marquardt-Levenberg method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Marquardt-Levenberg optimization method. The optimization

procedure was run to obtain the optimum parametric values. Figure 6.21 shows the variation of the sum of squared deviations of the random subset with the iterations. The number of iterations took to obtain the optimum values of the parameters, is clearly indicated in the figure. The objective function values that resulted from the latter set of parameter values were compared to that obtained from the former using the mean sum of squared distances. The mean sum of squares distances of the data points from the 'x = y' line is shown in Table 6.21. The comparison plot is shown in Figure 6.22.

Table 6.22: Mean Sum of Squares Distances of Data Points from 'x = y' Line using Marquardt-Levenberg Method

Stopping Criteria	Mean Sum of Squared Distances
Excessive Iterations	0.00648
SS Technique	0.00814

From Table 6.21, it can be concluded that the mean sum of squared distances is almost equal for both criteria. The parametric values obtained for both the curves are listed below in Table 6.22.

Table 6.23: Parameter values for the reaction kinetic model using Marquardt-Levenberg method

Parameters	Excessive Iterations	Steady-State stopping Criterion
A (1/s)	9.4565 E+6	9.4673 E+6
E (J/mol)	5934.345	5968.357
k_b (1/s)	2.2486	2.5435

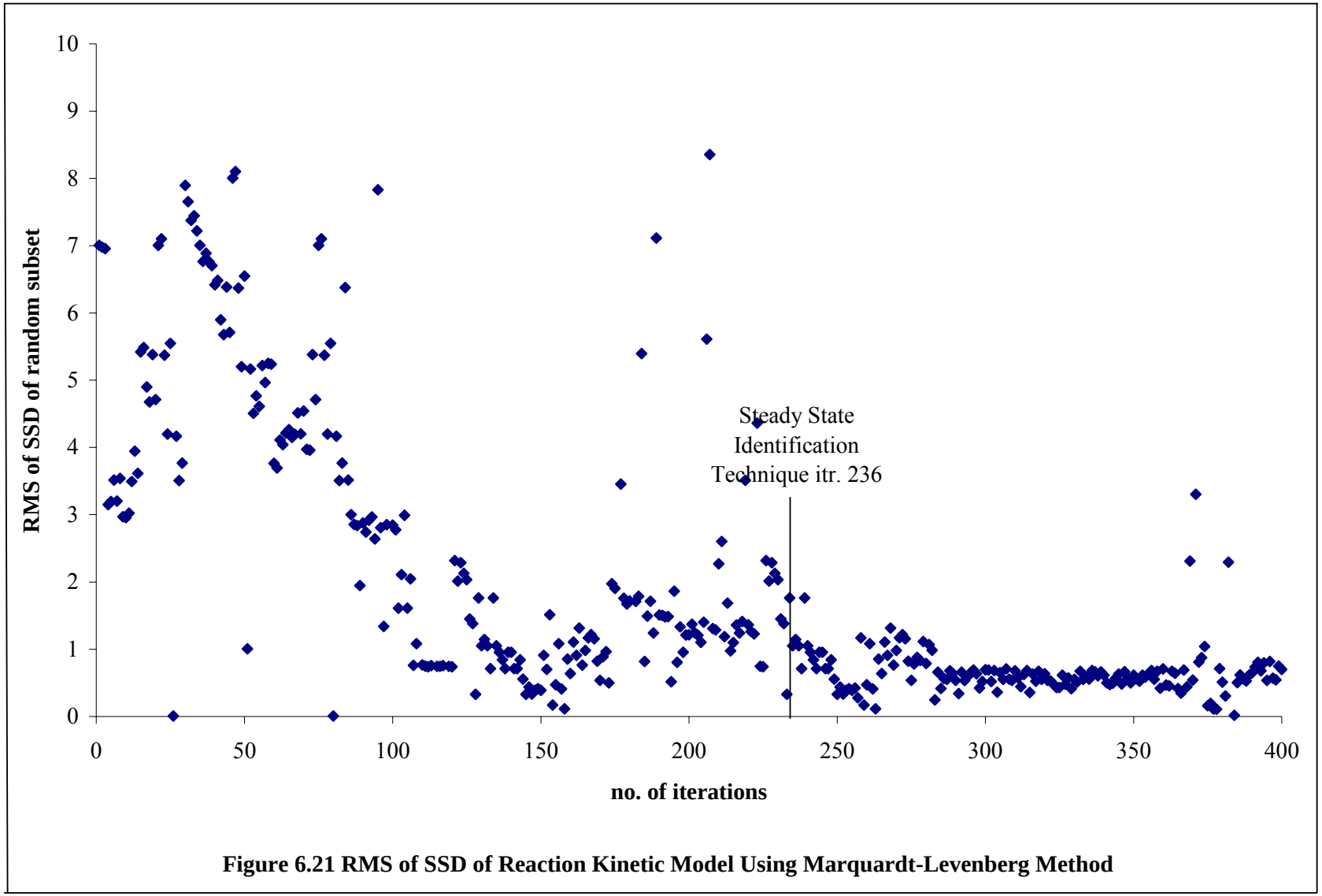


Figure 6.21 RMS of SSD of Reaction Kinetic Model Using Marquardt-Levenberg Method

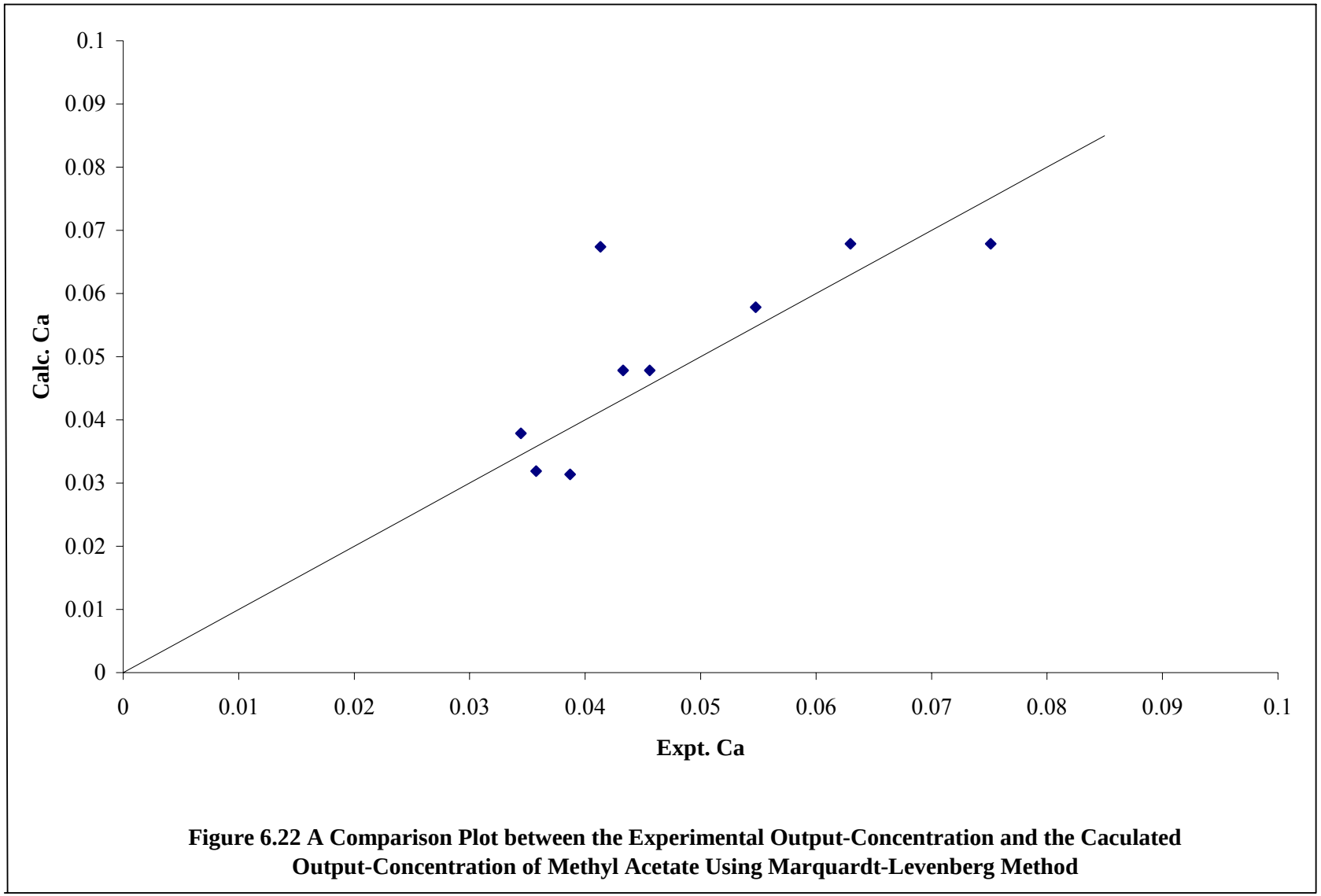


Figure 6.22 A Comparison Plot between the Experimental Output-Concentration and the Caculated Output-Concentration of Methyl Acetate Using Marquardt-Levenberg Method

Case 6.2.1.2 Optimization of parameters in the reaction kinetic model using Gauss-Newton method

The random number generation program in MATLAB 6.5 was again used to provide the initial guess to the Gauss-Newton optimization method. The optimization procedure was run for an excessive number of iterations until no change in the SSD of the random subset was observed, to obtain the optimum parametric values. Figure 6.23 shows the variation of the sum of squared deviations of the random subset with the iterations. The objective function values that resulted from the latter set of parameter values were compared to that obtained from the former using the mean sum of squared distances. The mean sum of squares distances of the data points from the 'x = y' line is shown in Table 6.23. The comparison plot is shown in Figure 6.24.

Table 6.24: Mean Sum of Squares Distances of Data Points from 'x = y' Line using Gauss-Newton Method

Stopping Criteria	**Mean Sum of Squared Distances**
Excessive Iterations	0.00875
SS Technique	0.00943

From Table 6.23, it can be concluded that both the mean sum of squared distances is almost equal for both criteria. The parametric values obtained for both the curves are listed below in Table 6.24.

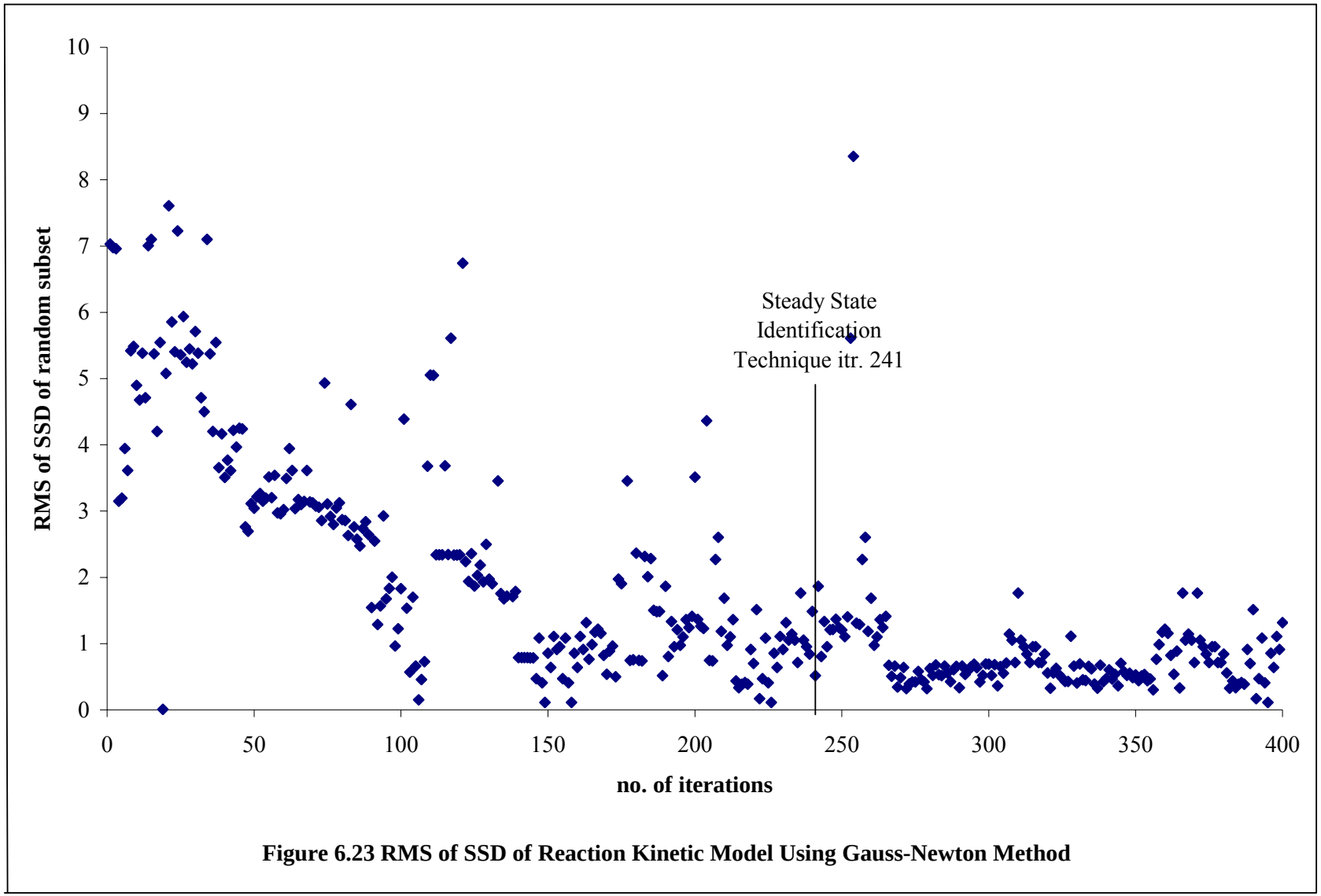


Figure 6.23 RMS of SSD of Reaction Kinetic Model Using Gauss-Newton Method

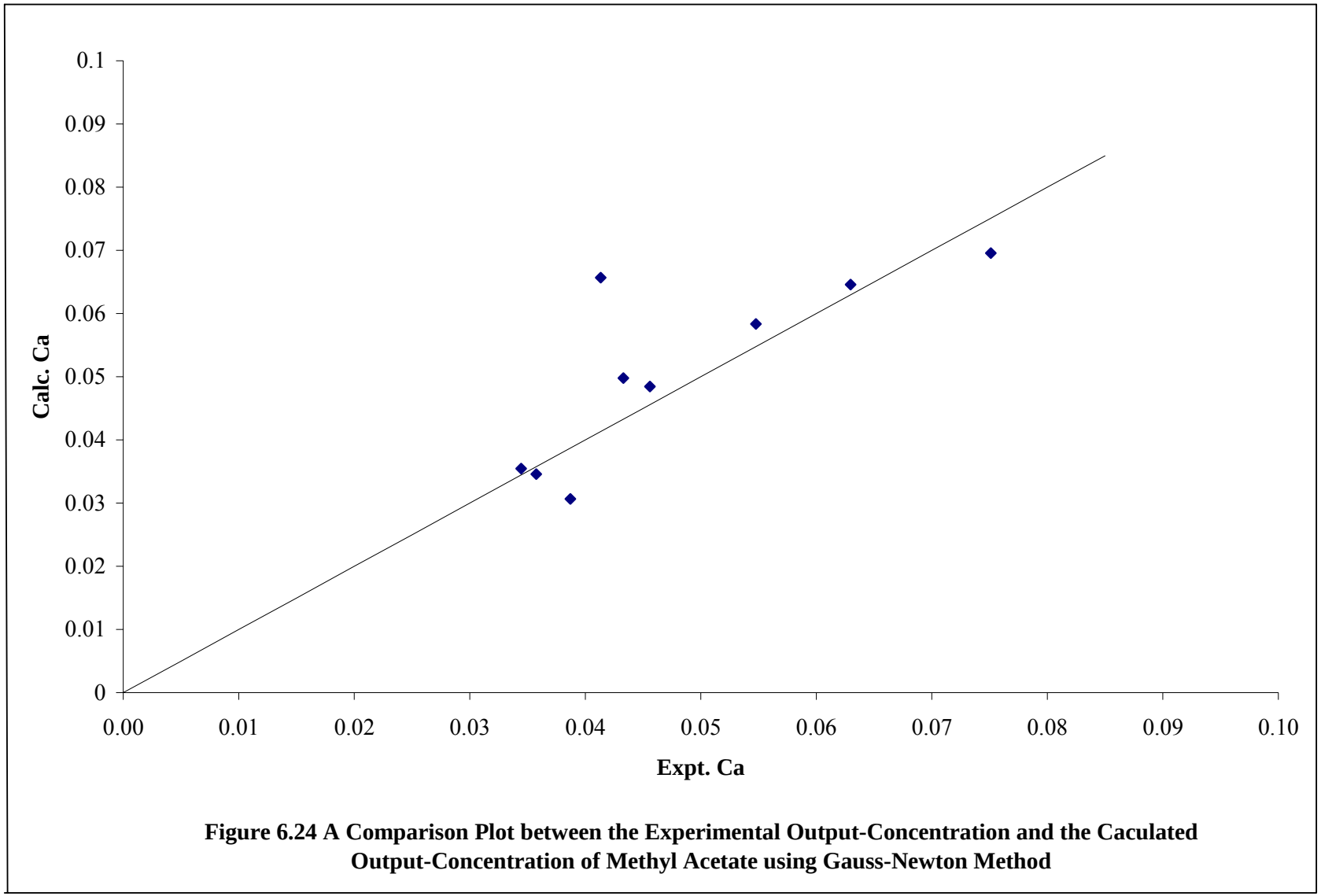


Figure 6.24 A Comparison Plot between the Experimental Output-Concentration and the Caculated Output-Concentration of Methyl Acetate using Gauss-Newton Method

Table 6.25: Parameter values for the reaction kinetic model using Gauss-Newton method

Parameters	Excessive Iterations	Steady-State stopping Criterion
A (1/s)	9.4223 E+6	9.4251 E+6
E (J/mol)	5814.687	5986.542
k_b (1/s)	2.3557	2.7381

6.2.2 Optimization of Parameters in a Two-Phase Flow Equation

The two-phase flow is a simultaneous flow of both gas and liquid phase fluids through a pipe or a tube. This phenomenon occurs extensively in chemical engineering unit operations such as distillation columns, evaporators, gas pipelines, condensers, reactors etc. The experimental setup consists of a long vertical glass pipe through which the liquid and the gas flow. The fluid flow rates are monitored using rotameters in coordination with orifice meters and the Camile software is used to control them. Pressure transducers measure the pressure at both the top and the bottom of the vertical column. The experimental data are shown in Appendix B.

Several methods are used to analyze the two phase flow. In this experimental study, the pressure drop per unit length in two phase flow systems is calculated from the Lockhart-Martinelli correlation. It is then compared with the experimental values. A sample calculation for the pressure drop is shown in Appendix C.

The four Lockhart-Martinelli correlation constants, C are obtained from the literature [19]. For different flow patterns, the iterative values of C are given in Table 6.26.

The fluid flow is laminar or turbulent depending on the Reynold's number. The classification of flow based on the Re values is given in Table 6.25.

Table 6.26: Flow patterns of fluid based on Reynold's number

Flow Pattern	Reynold's Number
Laminar	Re < 2000
Turbulent	3000 < Re < 50000

Table 6.27: Lockhart-Martinelli correlation constant for different vapor-liquid flow patterns

Liquid	Vapor	C
Laminar	Laminar	5
Turbulent	Laminar	10
Laminar	Turbulent	12
Turbulent	Turbulent	20

It is evident from the above table that the value of C is dependent on the Reynolds's number of both the liquid and the vapor. An effort was put in to obtain more accurate values for the correlation constant, C, by choosing a model that involves both the gas and liquid Reynolds's numbers. The model selected is given by Equation 6.6.

$$C_i = a_i \, \mathrm{Re}_l^{\,b_i} \, \mathrm{Re}_g^{\,c_i} \qquad (6.6)$$

The three coefficients a, b, and c, for each of the four laminar-turbulent cases were the DV in the optimization to make the Lockhart-Martinelli model best predict the experimentally measured pressure drop from these experiments. The data was classified into four groups depending on the flow patterns of the gas and the liquid. Two sets of the parameter values were obtained for each case while optimizing with the conventional stopping criterion with excessive iterations and the novel stopping criterion using the steady state identification technique. The results obtained using the Nelder-Mead Simplex method is shown below. The values of the parameters were then used to evaluate the values of the constant which in turn were used to calculate the pressure drop. The pressure drops thus obtained were compared with the experimental values as indicated by the Camile software. The comparison plots are shown in Figure 6.25, Figure 6.26 and Figure 6.27. The classification and the results obtained in each are discussed in the cases below.

Case 6.2.2.1 Liquid Flow - Laminar

Gas Flow - Laminar

The values of a, b, and c for this case of Laminar-Laminar flow is given in Table 6.27.

Table 6.28: Parameter values for the model equation and the C value for Laminar-Laminar flow patterns of liquid and gas respectively

Constants	Excessive Iterations	Steady-State stopping Criterion
a	7.4918	7.5849
b	0.7035	0.6143
c	-0.7573	-0.6919

Case 6.2.2.2 Liquid Flow - Turbulent

Gas Flow - Laminar

The values of a, b, and c for this case of Turbulent -Laminar flow is given in Table 6.28.

Table 6.29: Parameter values for the model equation and the C value for Turbulent -Laminar flow patterns of liquid and gas respectively

Constants	Excessive Iterations	Steady-State stopping Criterion
a	0.32065	0.26464
b	0.8173	0.7549
c	-0.4553	-0.3664

Case 6.2.2.3 Liquid Flow - Turbulent

Gas Flow - Turbulent

The values of a, b, and c for this case of Turbulent - Turbulent flow is given in Table 6.29.

Table 6.30: Parameter values for the model equation and the C value for Turbulent - Turbulent flow patterns of liquid and gas respectively

Constants	Excessive Iterations	Steady-State stopping Criterion
a	20.4434	19.1364
b	0.7394	0.76464
c	-0.7255	-0.74646

Case 6.2.2.4 Liquid Flow - Laminar

Gas Flow - Turbulent

The values of a, b, and c for this case of Laminar - Turbulent flow is given in Table 6.30.

Table 6.31: Parameter values for the model equation and the C value for Laminar - Turbulent flow patterns of liquid and gas respectively

Constants	Excessive Iterations	Steady-State stopping Criterion
a	5.0199	4.87646
b	0.9122	0.8664
c	-0.6254	-0.5944

The SSD of the data points from the 'x = y' line is shown in Table 6.31.

Table 6.32: The average SSD of the data points from the 'x = y' line

	C from Literature	C from Excessive Iterations	C from Steady-State stopping Criterion
SSD/(N-1)*	0.13240	0.11644	0.12421

*N is the number of data points, N = 65.

Table 6.30 gives the average distance of all the data points from the 'x = y' line. It is evident from Table 6.31 that the deviation of the data points from the line in all the three cases is almost the same.

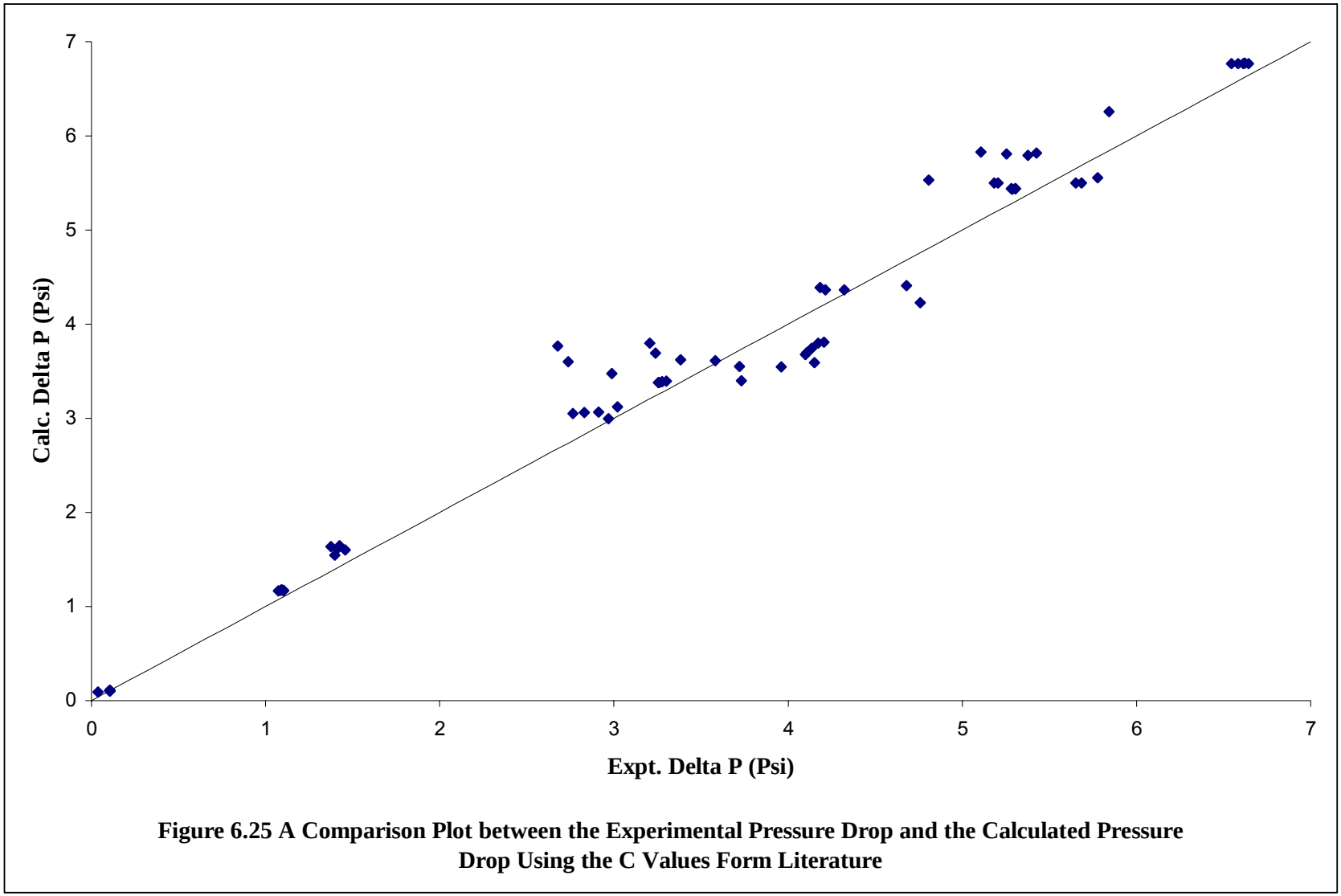


Figure 6.25 A Comparison Plot between the Experimental Pressure Drop and the Calculated Pressure Drop Using the C Values Form Literature

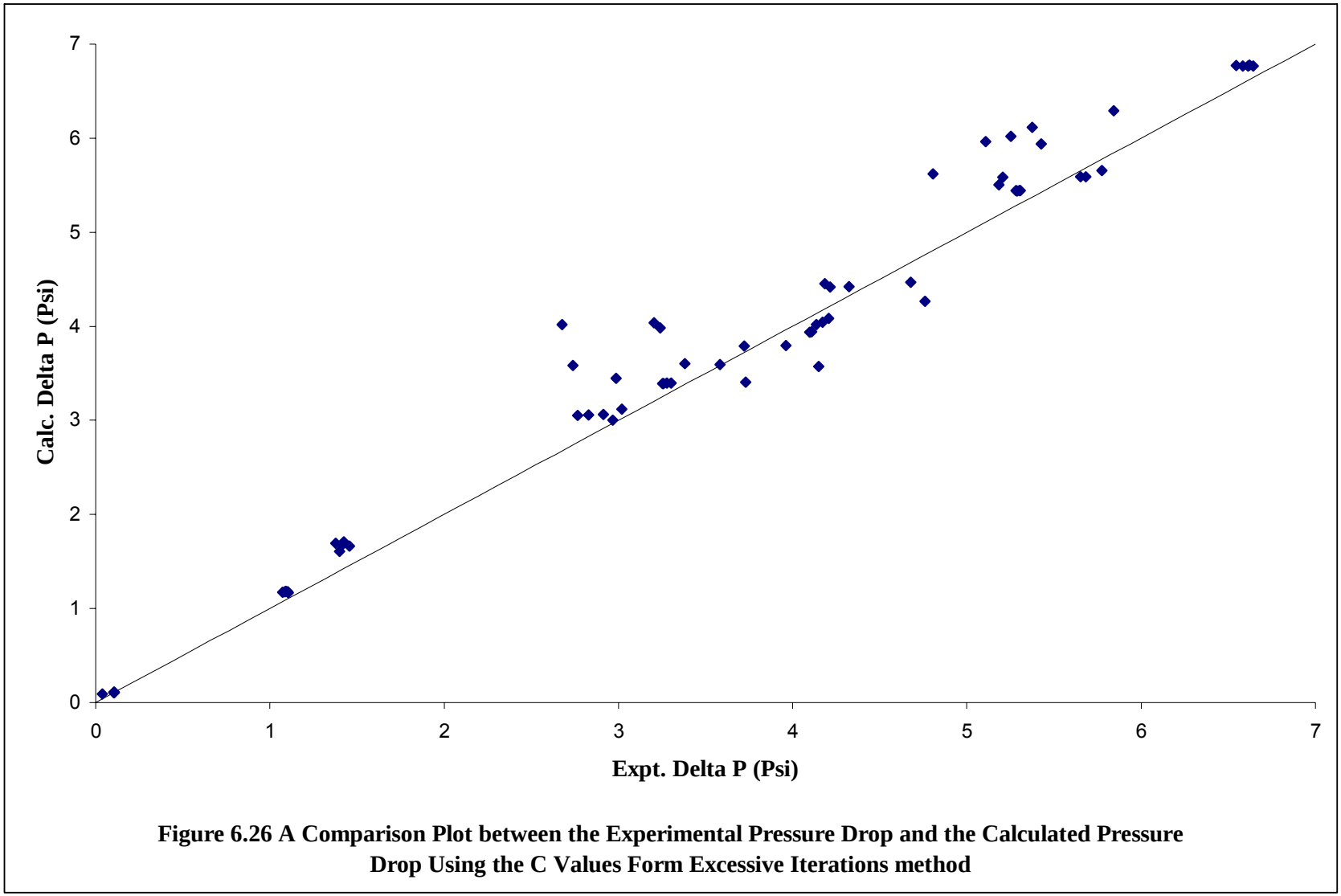


Figure 6.26 A Comparison Plot between the Experimental Pressure Drop and the Calculated Pressure Drop Using the C Values Form Excessive Iterations method

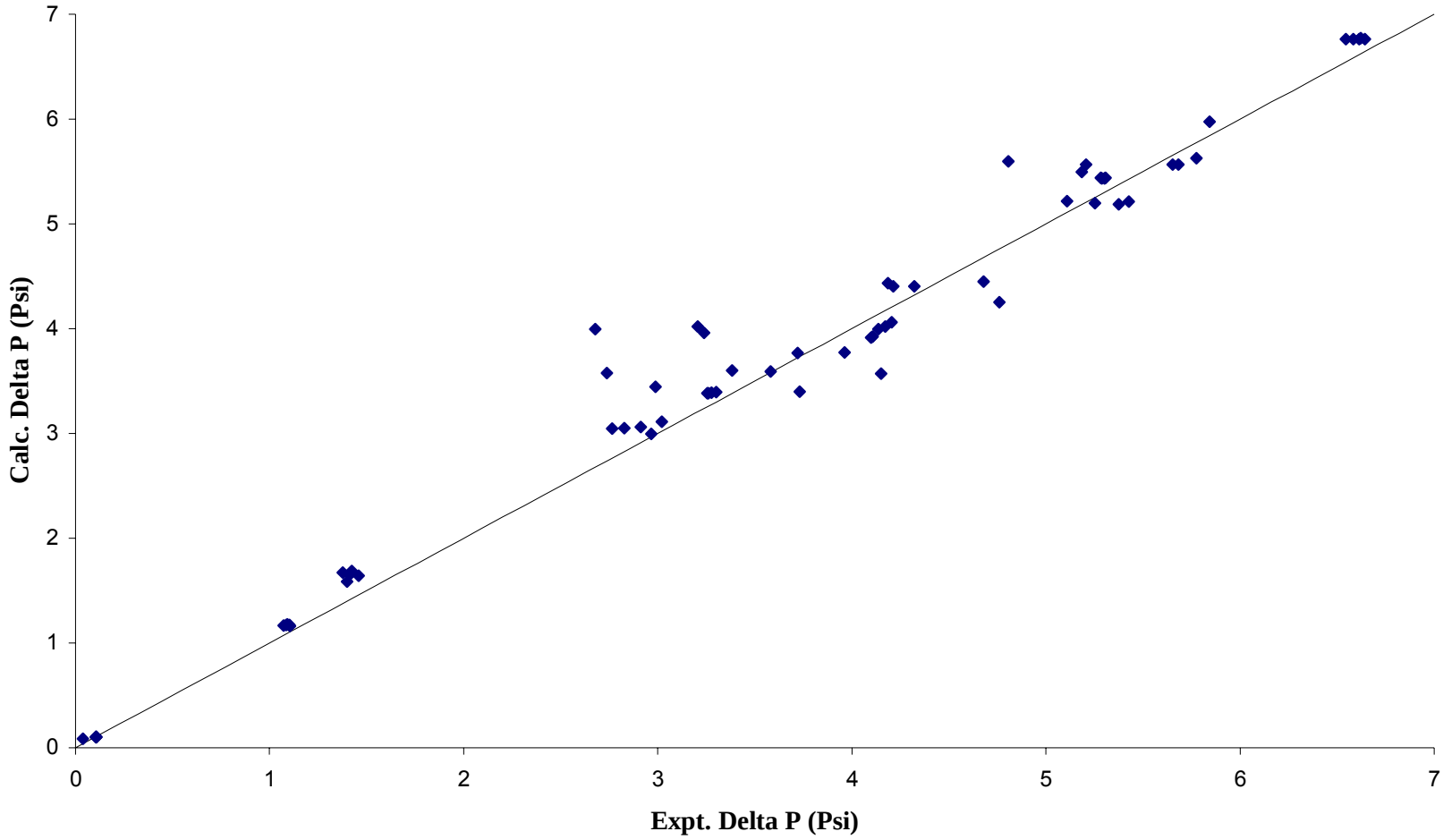


Figure 6.27 A Comparison Plot between the Experimental Pressure Drop and the Calculated Pressure Drop Using the C Values Form Steady State Technique

6.3 Discussion

In an earlier investigation [20], this technique was used as the stopping criterion for both the Levenberg-Marquardt and error back propagation methods for neural network training. While the number of decision variables (15 to 30 weights) was larger than the number in this work (2 model coefficients), the application was of one type. While all problems in this work were low dimensional, this work extends the applications and optimizations to demonstrate the practicality of this steady state stopping criterion on a wider variety of problems [20].

The novel stop-training technique was used to stop neural network training [21] when the improvement in prediction is insignificant relative to the variability in the residuals. Based on their technique, after each epoch, 20 to 30% of the data set was randomly selected. This is the validation set for that particular epoch. Each epoch will have a unique validation set. The RMS error on the validation set was computed, and was plotted against epoch number. As the number of training epochs increase, the plot will asymptotically approach a low value [20]. However, the curve will be a "noisy" reflection of the random choice of the validation set. When there is no visible improvement in training (when the change in RMS value is small relative to the noise on the RMS value), it is stopped. This was easily done visually, as if declaring when a noisy variable reaches steady-state. Else, any one of a number of automated steady-state identification techniques could be used. Both the visual and automated steady-state stop-training trigger (SSSTT) approaches were explored on a variety of applications and compared with conventional practice.

Since these optimization applications were of low dimension, the optimization approaches immediately started "down hill" to minimize the Objective Function value. By contrast, in the prior work with many decision variables, the improvement in the OF value in the initial iterations was often slight, and the plot of random subset SSD with respect to iteration number would appear to be at steady state initially. This would stop the optimization prior to making progress. Consequently, the broader, two-condition rule, "Stop optimization when steady state is identified subsequent to a transient period." Was unnecessary for this work. That additional logic would not affect the results.

The comparison of this steady state stop optimization criterion to the conventional operator-decision based on cross validation in training neural networks concluded that the automated method gave equivalent RMS values and chose to stop with less iteration [22]. The automation advantage of this method was subsequently used in evaluating the probability of finding a global minimum in training thousands of neural networks. This work supports that finding on a variety of conventional applications.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

The novel stopping criterion for optimization, based on identifying steady state of a random subset of the sum of squared deviations with respect to iteration number, was formerly explored for neural network training. In this work, it has been extended to demonstrate advantages on a variety of empirical modeling optimization applications.

The novel stop-optimization criterion was tested on a different variety of applications involving various kinds of objective functions. On all the cases, the novel stop-optimization criterion gives equivalent results (as measured by model residuals) to the best possible results, with a sufficient (not excessive) number of iterations and without *a priori* knowledge of the optimization problem (scale, end-point values, and other classic stopping criteria).

The method is ready for commercial use and hence, the recommendation would be to spread out the word to the world so that this novel stop-iteration technique could be implemented in the optimization softwares developed.

BIBLIOGRAPHY

1. K. Lange, “Optimization”, Springer-Verlag NY, LLC, 2004
2. R.S. Anderssen, L.S. Jennings and D.M. Ryan, “Optimization”, University of Queensland Press,1972
3. S. Chandran, “Heuristic Random Optimization”, A Thesis in Chemical Engineering, Texas Tech University, 1998
4. W.E Biles and J.J Swain, “Optimization and Industrial Experimentation”, Wiley Eastern Publications, c1980
5. A.V. Balakrishnan and M. Thoma, “Lecture Notes in Control and Information Sciences”, Springer-Verlag Co., NY, LLC, 1982
6. S. Cox and I. Lasiecka, “Optimization Methods In Partial Differential Equations”, *Proceedings of the 1996 Joint Summer Research Conference*, Contemporary mathematics (American Mathematical Society) , v. 209
7. G.D. Pillo and F. Giannessi, “Nonlinear Optimization and Applications”, Plenum Press, NY, 1996
8. R.W. Pike, “Optimization for Engineering Systems”, Van Nostrand Reinhold Co., NY, c1986
9. J.G. Rau, “Optimization and probability in systems engineering”, Van Nostrand Reinhold Co., NY, 1970
10. E. Minieka, “Optimization Algorithms for Networks and Graphs”, Industrial Engineering ; v. 1, M. Dekker Co., NY, c1978

11. V. Padmanabhan and R. Rhinehart, “A Novel Termination Criterion for Optimization”, *Proceedings of the 2005 American Control Conference*, June 8-10, 2005, pp 2281-2286
12. S. Cao and R. Rhinehart, “An Efficient Method for On-line Identification of Steady State”, J. Process Control; v. 5, No. 6, 1995, pp 363-374
13. P.R. Adby and M.A.H. Dempster, “Introduction to Optimization Methods”, Halsted Press, 1974
14. M.J.D. Powell, “Nonlinear Optimization”, Academic Press, NATO Scientific Affairs Division, New York, 1982
15. L.E Scales, “Introduction to Nonlinear Optimization”, Springer-Verlag NY, LLC, 1985
16. J.T Szela and R. Rhinehart, “A Virtual Employee to Trigger Experimental Conditions”, JPAC, Process Analytical Chemistry, pp 1-4
17. O. Levenspiel, “Chemical Reaction Engineering”, 3rd edition, Wiley Eastern Publications, NY, c1999
18. T.F. Edgar, D.M. Himmelblau and L.S. Lasdon, “Optimization of Chemical Processes”, second edition, McGrawHill Publications, 2001
19. J.B. Diaz, “Fluid dynamics and applied mathematics”, Gordon and Breach Co., NY, 1962
20. M./ Iyer and R. Rhinehart, “A Novel Method to Stop Neural Network Training”, *Proceedings of the 2000 American Control Conference*, v. 2, June 28-30, 2000 pp 929 - 933

21. S. Natarajan and R. R. Rhinehart, "Automated Stopping Criteria For Neural Network Training," *Proceedings of the 1997 American Control Conference,* Albuquerque, NM*,* June 1997, Paper # TP09-4
22. S. Cao and R. Rhinehart, "Critical Values for Steady State Identifier" J. Process Control, v. 7, 1997, pp 149-154

APPENDICES

APPENDIX A

CAMILE TG 4.0 SOFTWARE STARTUP AND OPERATIONS

A1 Startup

The following steps must be performed to open the Camile Software and create the necessary two-phase operation file.

1. Turn on computer.

   Log in (username and password required).

   Double click Camile TG 4.0 icon on the desktop.

2. Go to the file menu

   Open applications.

   Select C:\ drive.

   Select UOL Two-Phase flow directory.

   Double click Two-Phaseflow.app.

The following steps must be performed to create a new task name for any UOL operations and to select the variables that will be input into the results file.

1. Go to the task menu

   Select Logging.

   Select New.

   Enter username.

   Click OK.

2. Go to Edit menu

   Highlight your name.

   Set Logging rate (2 to 5 seconds recommended).

   Click Insert field.

   In Fields window, double click Software Tag.

   Add: bottom_Pressure

   Delta_Pressure

   Fl_1_Filt

   Fl_2_Filt

   Fl_3_Filt

   SSIP_1

   SSIT_1

   Top_Pressure

   Water_Height

   SC_1Valve

3. From the Fields window, click Camile Box. Select the following:

   Camile_1

   Box ID

   DIO_BOARD_10Mz_1

   SC_1Valve

   Click Add

Click OK

4. In Logging-Edit Template window, go to Format Drop box

   Click Text.

   Click OK.

The following steps must be performed to create a notepad file which will be used for data reports.

1. From Logging Configurations

   Click open.

   Enter new file name.

   Check Logging Slate – enabled.

   Check Status – overwrite.

   Click Done.

The following steps must be performed to begin a Camile two-phase run.

1. Go to Run drop box

   Click Start Run.

   Observe pop up window.

   Click OK.

   Click Two-Phase Flow Window Save.

2. From Logging_Overwrite Confirmation Window

Verify that username is highlighted.

Click OK.

A2 Camile TG 4.0: Using Virtual Employee

1. Go to the desktop

   Double click Camile TG 4.0 icon.

2. Go to File menu

   Open Applications.

   Open in C:\ drive.

   Select UOL two-phase flow folder.

   Double click Two-PhaseFlow.app.

   Click Run drop box.

   Click Start Run.

   Click OK.

   Click Save.

3. Make sure the results file is highlighted.

4. Press and hold CTRL and press TAB button.

   Keep doing so until Controls of Two Phase Flow Screen appears.

5. Click YELLOW BUTTON # STOP while holding down the CTRL button – BUTTON # START.

6. Camile will automatically run and record the data.

7. Press and hold CTRL and press TAB button.

8. Keep doing so until Graphical Data of Two Phase Flow screen appears.

9. Scroll down to the bottom of this screen to the Steady State Identification and All Measured Data Graph.

10. Make sure the system is at Steady State. The lines on the graph will be straight and horizontal for approximately one minute when the system is at steady state.

11. Go to Run drop box

    Click Stop Run.

12. Go to File drop box

    Click Exit Camile.

13. Program will ask if you want to save changes to Two-PhaseFlow.app.

    Click NO if no changes have been made.

Click YES if changes have been made.

14. Go to Start Button (bottom left corner of the screen)

    Click Program.

    Click Accessories.

    Double Click Notepad.

15. Click File drop box

    Click Open.

    Open the file that was saved to record the data.

APPENDIX B

EXPERIMENTAL DATA

B1 Data from the Packed Bed Reactor Experiment (PBR)

The experimental output-concentration of methyl acetate is obtained by titrating the sample collected with 0.164 M sodium hydroxide (NaOH) using phenolphthalein as indicator. The rate constant, ka is given by the Arrhenius equation given in Equation (i).

$$k_a = A_o e^{-\frac{E}{RT}} \tag{i}$$

The model that was used to optimize the values for the parameters, A, E and kb, is given in Equation (ii).

$$f(C_a) = L + \frac{F}{X_c A e^{-\frac{E}{RT}}}\left(\ln\left(\frac{C_a}{C_o}\right) + k_b(C_a - C_o)\right) = 0 \tag{ii}$$

Where,

L	-	Length of the reactor
F	-	Flow rate of reactants
X	-	Area of cross section of the reactor
A	-	Frequency factor
E	-	Activation energy
C_a	-	Output-concentration of methyl acetate
C_o	-	Input-concentration of methyl acetate
k_b	-	Rate constant

T - Temperature

R - Universal gas constant

The experimental data is shown in Table B1.

Table B1: Experimental Data from the Packed Bed Reactor

s. no.	T (C)	T (K)	Co (mol/L)	V sample (ml)	V NaOH (ml)	n NaOH (mol)	n sample (mol)	Ca (mol/L)
1	32	305.15	1.06	50	10.5	1.722	1.722	0.03444
2	32	305.15	1.328	50	10.9	1.7876	1.7876	0.035752
3	32	305.15	1.98	50	11.8	1.9352	1.9352	0.038704
4	37.4	310.55	1.06	50	12.6	2.0664	2.0664	0.041328
5	37.4	310.55	1.328	50	13.2	2.1648	2.1648	0.043296
6	37.4	310.55	1.98	50	13.9	2.2796	2.2796	0.045592
7	44.3	317.45	1.06	50	16.7	2.7388	2.7388	0.054776
8	44.3	317.45	1.328	50	19.2	3.1488	3.1488	0.062976
9	44.3	317.45	1.98	50	22.9	3.7556	3.7556	0.075112

B2 Data from the Two-Phase Flow Experiment

The results from the Two-phase Flow experiment are shown in Table B2.

Table B2: Experimental Data from the Packed Bed Reactor

S. no.	Delta_Pr.	large air flow FI_1_Filt (ft^3/min)	small air flow FI_2_Filt (ft^3/min)	liquid flow rate FI_3_Filt (kg/hr)	Water Ht. (m) W_Ht_Filt
1	0.0507	1.3498	0.051	91.1077	0.0334
2	0.0688	1.5193	0.0516	92.5997	0.0371
3	0.0479	1.5942	0.0475	90.4832	0.0334
4	0.0515	1.6495	0.0509	92.2022	0.0335
5	0.0381	1.668	0.0513	89.0564	0.0244
6	4.3754	24.9847	0.0544	519.5394	3.0113
7	4.2531	24.9838	0.0537	520.492	3.0191
8	4.1076	24.976	0.0518	513.5342	3.0412
9	4.4256	24.9957	0.0557	525.4731	3.021
10	6.6422	12.1766	0.0635	295.2589	3.803
11	6.6132	1.331	0.0477	88.4521	4.6495
12	6.546	1.3907	0.0479	88.3032	4.6502
13	6.6422	1.626	0.0499	90.5244	4.6483
14	6.5842	1.5733	0.0496	89.6316	4.6484
15	6.6224	1.3945	0.0481	89.0713	4.6542
16	4.1848	1.5374	1.0012	497.4645	2.8596
17	4.6788	1.6143	1.0012	496.4514	2.8711
18	4.2142	1.6453	1.0011	494.1385	2.8379
19	5.3232	1.5262	1.001	484.4626	2.8457
20	3.7591	1.4381	0.7254	417.8737	2.7753

Table B2 (contd.)

S. no.	Delta_Pr.	Large air flow Fl_1_Filt (ft^3/min)	Small air flow Fl_2_Filt (ft^3/min)	Liquid flow rate Fl_3_Filt (kg/hr)	Water Ht. (m) W_Ht_Filt
21	3.0983	6.9099	0.0535	506.7354	2.0248
22	3.1356	6.9465	0.0548	518.5857	2.0634
23	2.2391	6.9189	0.0524	526.2402	2.0251
24	3.2055	6.8981	0.0526	514.9471	2.1135
25	2.6775	7.039	0.0532	505.1873	2.0855
26	4.8066	1.4871	0.5015	516.2134	3.6626
27	5.6527	1.5952	0.5011	517.7882	3.6403
28	5.1833	1.6012	0.501	517.1487	3.6385
29	4.7757	1.394	0.5	519.9964	3.6841
30	5.2051	1.7096	0.5006	515.911	3.6371
31	5.2872	1.4949	0.0504	102.3022	3.7237
32	5.3022	1.5885	0.0518	101.6442	3.7253
33	5.2826	1.5451	0.0517	101.3847	3.7269
34	5.306	1.4748	0.0511	101.0812	3.7267
35	5.1835	1.6051	0.0503	100.5808	3.7672
36	3.02	1.566	0.5015	99.8789	2.1164
37	2.913	1.3943	0.5012	99.995	2.0793
38	2.8286	1.5553	0.5014	99.965	2.0742
39	2.7656	1.3057	0.5004	99.8528	2.0707
40	2.9677	1.3522	0.1949	100.9842	2.0342
41	3.2543	1.2259	0.048	99.8881	2.3033
42	3.2596	1.2069	0.0498	100.2032	2.304
43	3.3017	1.4645	0.0517	101.5342	2.3096
44	3.2772	1.2805	0.0473	99.978	2.3085

Table B2 (contd.)

S. no.	Delta_Pr.	Large air flow FI_1_Filt (ft^3/min)	Small air flow FI_2_Filt (ft^3/min)	Liquid flow rate FI_3_Filt (kg/hr)	Water Ht. (m) W_Ht_Filt
45	3.7305	1.4216	0.0521	101.1915	2.3161
46	4.1507	1.3397	1.0008	297.1745	2.3546
47	2.8382	1.329	1.0005	299.4958	2.3747
48	3.6234	1.392	1.0002	298.5453	2.3681
49	2.7378	1.5377	1.0003	304.526	2.3544
50	2.9877	1.3748	0.6925	241.8319	2.2926
51	3.1209	7.0204	0.0509	496.462	1.9407
52	2.7202	7.088	0.0537	506.15	2.1037
53	3.2052	7.0773	0.0531	502.9768	2.1059
54	1.9616	6.9554	0.0512	502.3859	1.9336
55	3.1072	7.0119	0.0547	504.0659	2.036
56	1.4118	7.0717	0.051	99.3628	0.9594
57	1.4246	6.9956	0.0514	100.1572	0.9835
58	1.3984	6.928	0.0481	98.4197	0.9171
59	1.3768	6.9778	0.0514	99.0677	0.9765
60	1.4579	6.9697	0.0499	98.8602	0.9545
61	1.103	1.5693	0.0492	99.2359	0.7712
62	1.1041	1.742	0.0491	98.4339	0.7674
63	1.0864	1.4532	0.0503	98.4255	0.7726
64	1.0923	1.4741	0.0511	98.3083	0.776
65	1.0722	1.4127	0.0511	100.2193	0.768

APPENDIX C

SAMPLE CALCULATIONS

C1 Sample Calculations for the Novel Stopping Criterion

Table C1: Sample Calculations for the Novel Stop-Iterations Technique

λ_1 = 0.1 λ_3 = 0.05

λ_2 = 0.2

Itr.	SSD	RMS SSD	x_f	$v^2_{f,i}$	$\delta^2_{f,i}$	R
1	3.29E+03	2.57E+01	2.57E+01	0	0	
2	2.27E+03	2.13E+01	25.22457	3.807115	0.951779	7.6
3	3.23E+03	2.54E+01	25.24184	3.051656	1.744426	3.323813
4	3.31E+03	2.57E+01	25.28965	2.487059	1.662414	2.842499
5	1.90E+03	1.95E+01	24.70866	8.740685	3.52636	4.709475
6	1.79E+03	1.89E+01	24.12724	13.75358	3.36717	7.760759
7	2.27E+03	2.13E+01	23.84754	12.56748	3.495472	6.831185
8	1.08E+03	1.47E+01	22.92962	26.90556	5.539761	9.227936
9	302.6909	7.78E+00	21.41472	67.42285	7.634797	16.77889
10	68.6745	3.71E+00	19.64386	116.6576	8.083162	27.42113
11	296.2818	7.70E+00	18.44925	121.8677	8.475708	27.31908
12	37.7308	2.75E+00	16.87903	146.8061	9.277438	30.06559
13	1.0414	4.56E-01	15.23676	171.3856	9.07592	35.87875
14	23.8374	2.18E+00	13.93143	171.1862	8.771264	37.08176
15	7.1264	1.19E+00	12.65768	169.3982	8.381667	38.40007

Table C1 (Contd.)

Itr.	SSD	RMS SSD	x_f	$v^2_{f,i}$	$\delta^2_{f,i}$	R
16	2.3561	6.86E-01	11.46055	164.1806	7.975456	39.11289
17	0.2019	2.01E-01	10.33459	156.7002	7.588469	39.23458
18	0.339	2.60E-01	9.327172	145.6581	7.209222	38.38839
19	5.3398	1.03E+00	8.497797	130.2837	6.87864	35.98663
20	0.3605	2.69E-01	7.674869	117.7712	6.563962	34.08997
21	36.1845	2.69E+00	7.176397	99.18645	6.52898	28.86427
22	1.0914	4.67E-01	6.505477	88.35182	6.449605	26.02771
23	0.241	2.20E-01	5.876884	78.58404	6.130191	24.35645
24	11.0075	1.48E+00	5.43757	66.72717	5.903592	21.47534
25	0.6669	3.65E-01	4.930334	58.5275	5.670968	19.60904
26	0.145	1.70E-01	4.45433	51.35359	5.389319	18.10467
27	3.0312	7.79E-01	4.086759	43.78505	5.138356	16.19031
28	3.4712	8.33E-01	3.761404	37.14516	4.881587	14.45755
29	0.9074	4.26E-01	3.427864	31.9411	4.645799	13.06301
30	4.6672	9.66E-01	3.181692	26.76489	4.428097	11.48423

C2 Sample Calculations for Pressure Drop in Two-Phase Flow Apparatus Using Lockhart-Martinelli correlations

Density of Air

The density of air can be found out from the atmospheric pressure (P), its molecular weight (MW), the gas constant (R), and temperature (T):

$$\rho_g = \frac{MWP_{avg}}{RT_{avg}} \qquad \text{(i)}$$

$$\rho_g = \frac{24.9 \frac{lb_m}{lbmol} * 742.2 mmHg}{998.9 \frac{mmHg.ft^3}{lbmol.K} * 293.15 K} = 0.06313 \frac{lb_m}{ft^3} = 1.0135 \frac{kg}{m^3}$$

Density of Water

$$\rho_l = 28.282 \frac{kg}{ft^3} = 998.77 \frac{kg}{m^3}$$

The void fraction is obtained from the following equation. The values of h_v and h are given by the Camile output data.

$$\varepsilon_g = \frac{Vol_g}{Vol_l} = \frac{h_v}{h} \quad \text{(ii)}$$

$$\varepsilon_g = \frac{2.6021m}{5.44m} = 0.4783$$

$$\rho_{TP} = \varepsilon_g . \rho_g + (1 - \varepsilon_g) . \rho_l \quad \text{(iii)}$$

$$\rho_{TP} = 0.4783 * 1.0135 \frac{kg}{m^3} + (1 - 0.4783) * 998.77 \frac{kg}{m^3} = 521.5133 \frac{kg}{m^3}$$

Reynold's Number

$$\mathrm{Re}_l = \frac{D \dot{m}_l}{A \mu_l} \quad \text{(iv)}$$

$$\mathrm{Re}_l = \frac{0.026m * 0.1372 \frac{kg}{s}}{5.57E-04m^2 * 0.00109 \frac{kg}{ms}} = 5878.1117$$

$$\mathrm{Re}_g = \frac{D \dot{m}_g}{A \mu_g} \quad \text{(v)}$$

$$\mathrm{Re}_g = \frac{0.026m * 0.00123\frac{kg}{s}}{5.57E-04m^2 * 3.23E-05\frac{kg}{ms}} = 1780.3518$$

Liquid is in turbulent flow and the gas is in laminar flow. Hence, the Lockhart-Martinelli constant is given by the following equation.

$$C_i = a_i \,\mathrm{Re}_l^{\,b_i}\,\mathrm{Re}_g^{\,c_i} \qquad \text{(vi)}$$

$$C = 0.26464 * 5878.1117^{0.7549} * 1780.3518^{-0.3664} = 11.9417$$

Mass fraction, x_g

$$x_g = \frac{\dot{m}_g}{\dot{m}_l + \dot{m}_g} = \frac{0.00123}{0.00123 + 0.1372} = 0.00889$$

Friction factor, *f*

$$f_l = \frac{64}{\mathrm{Re}_l} = \frac{64}{5878.1117} = 0.01088$$

$$f_g = \frac{64}{\mathrm{Re}_g} = \frac{64}{1780.3518} = 0.03594$$

$$X^2 = \frac{\left(\frac{\Delta P_f}{L}\right)_l}{\left(\frac{\Delta P_f}{L}\right)_g} = \frac{f_l\left(1-x_g\right)^2 \rho_g}{f_g\left(x_g\right)^2 \rho_l} \qquad \text{(vii)}$$

$$X^2 = \frac{0.01088 * \left(1-0.00889\right)^2 * 1.0135}{0.03594 * \left(0.00889\right)^2 * 998.77} = 3.8130$$

$$X = 1.9526$$

$$\left(\phi_g\right)^2 = 1 + C.X + X^2 \qquad \text{(viii)}$$

$$\left(\phi_g\right)^2 = 1 + 11.9417 * 1.9526 + 3.8130 = 28.3399$$

$$-\left(\frac{\Delta P_f}{L}\right)_g = \frac{2.f_g.\left(\frac{\dot{m}}{A}\right)^2.\left(x_g\right)^2}{\rho_g.D} = \frac{2 * 0.03594 * \left(\frac{0.00123}{5.57E-04}\right)^2 * 0.00889^2}{1.0135 * 0.026} = 13.3550\frac{Pa}{m}$$

$$\Delta P = \rho_{TP}.g = 521.5133 * 9.8 = 5204.7034\frac{Pa}{m}$$

$$\left(\frac{\Delta P_f}{L}\right)_{TP} = (\phi_g)^2 . \left(\frac{\Delta P_f}{L}\right)_g = 28.3399 * 13.3550 = 375.699 \frac{Pa}{m}$$

$$-\left(\frac{\Delta P}{L}\right) = 5204.7034 + 375.699 = 5580.4029 \frac{Pa}{m}$$

$$-(\Delta P) = 5580.4029 \frac{Pa}{m} * 5.44m = 30357.3922 Pa = 4.4018 Psi$$

APPENDIX D

MATLAB CODES FOR DIFFERENT OPTIMIZATION TECHNIQUES

D1 Linear model - Nelder-Mead Simplex method

Main Program

```
clear all;
clc;
% Generating random values of 'x' and 'y'.
x=0.01:0.5:10;
y=0.5.*x+0.2+0.4.*randn(size(x));
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter');
% Optimizing parameters
Estimates=fminsearch(@linsimf,Starting,options,x,y)
% To check the fit
clf;
plot(x,y,'*')
hold on
plot(x,(Estimates(1).*x+Estimates(2)),'r')
```

Subroutine

```
function sse=linsimf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(20,1).*20.+0.5);
r=o(1:5');
d=0.01+(r-1).*0.5;
% Defining parameters.
A=params(1);
B=params(2);
fc1=(A.*input+B);
% Calculating 'y' values for corresponding 'x' values.
fc2=(A.*d+B)';
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if fc2(i)==fc1(j)
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)

 h(i) = fc2(i) - Actual_Output(k(i));
```

```
end
% SSD calculation.
ssd=sum(h(i).^2);
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D2 Linear Model – Marquardt-Levenberg Method

Main Program

```
clear all;
clc;
% Define the data sets that you are trying to fit the
% function to
x=0.01:0.5:10;
y=0.5.*x+0.2+3.*randn(size(x));

% Initialize the coefficients of the function
%X0=[1 1 1 1 1]';
%
%
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter','Largescale','off','LevenbergMarquard
t','on','maxFunEvals',400);
%
%
% Set an options file for LSQNONLIN to use the
% medium-scale algorithm
%options = optimset('Largescale','off');

% Calculate the new coefficients using LSQNONLIN
Estimates=lsqnonlin(@linmlf,Starting,[],[],options,x,y);

% Plot the original and experimental data
clf;
plot(x,y,'*')
hold on
plot(x,(Estimates(1).*x+Estimates(2)),'r')
```

Subroutine

```
function Error_Vector=linmlf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(20,1).*20.+0.5);
r=o(1:5');
d=0.01+(r-1).*0.5;
% Defining parameters.
A=params(1)
B=params(2)
fc1=(A.*input+B);
```

```
% Calculating 'y' values for corresponding 'x' values.
fc2=(A.*d+B)';
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if fc2(i)==fc1(j)
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2)
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D3 Linear Model – Gauss-Newton Method

Main Program

```
clear all;
clc;
% Define the data sets that you are trying to fit the
% function to
x=0.01:0.5:10;
y=0.5.*x+0.2+3.*randn(size(x));

% Initialize the coefficients of the function
%X0=[1 1 1 1 1]';
%
%
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter','Largescale','on','LevenbergMarquardt
','off','maxFunEvals',400);
%
%
% Set an options file for LSQNONLIN to use the
% medium-scale algorithm
%options = optimset('Largescale','on');

% Calculate the new coefficients using LSQNONLIN
Estimates=lsqnonlin(@linmlf,Starting,[],[],options,x,y);

% Plot the original and experimental data
clf;
plot(x,y,'*')
hold on
plot(x,(Estimates(1).*x+Estimates(2)),'r')
```

Subroutine

```
function Error_Vector=linmlf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(20,1).*20.+0.5);
r=o(1:5');
d=0.01+(r-1).*0.5;
% Defining parameters.
A=params(1)
B=params(2)
fc1=(A.*input+B);
% Calculating 'y' values for corresponding 'x' values.
fc2=(A.*d+B)';
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if fc2(i)==fc1(j)
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2)
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D4 Nonlinear Model – Nelder-Mead Simplex Method

Main Program

```
clear all;
clc;
channel = ddeinit('excel','data for packed bed reactor.xls');
% Generating random values of 'x' and 'y'.
x=273:500:19773;
y=5.*log(55.*x)+3.*randn(size(x));
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter');
rc = ddepoke(channel, 'r3c1:r42c1', x);
rc = ddepoke(channel, 'r3c2:r42c2', y);
% Optimizing parameters
Estimates=fminsearch(@nonlinsimf,Starting,options,x,y)
% To check the fit
clf;
plot(x,y,'*')
hold on
plot(x,Estimates(1).*log(Estimates(2).*x),'r')
```

Subroutine

```
function sse=nonlinf(params,input,Actual_Output)
channel = ddeinit('excel','data for packed bed reactor.xls');
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(40,1).*40.+0.5);
r=o(1:5');
d=273+(r-1).*500;
% Defining parameters.
A=params(1);
lamda=params(2);
fc1=A.*log(lamda.*input);
% Calculating 'y' values for corresponding 'x' values.
fc2=A.*log(lamda.*d);
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if(fc2(i)==fc1(j))
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2);
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
rc = ddepoke(channel, 'r3c3', A);
rc = ddepoke(channel, 'r3c4', lamda);
rc = ddepoke(channel, 'r3c7', ssd);
```

D5 Nonlinear Model – Marquardt-Levenberg Method

Main Program

```
clear all;
clc;
% Define the data sets that you are trying to fit the
% function to
x=273:500:19773;
y=5.*log(55.*x)+2.*randn(size(x));

% Initialize the coefficients of the function
%X0=[1 1 1 1 1]';
%
%
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter','Largescale','off','LevenbergMarquard
t','on','maxFunEvals',400);
%
```

```
% Set an options file for LSQNONLIN to use the
% medium-scale algorithm
%options = optimset('Largescale','off');

% Calculate the new coefficients using LSQNONLIN
Estimates=lsqnonlin(@nonlinmlf,Starting,[],[],options,x,y);

% Plot the original and experimental data
clf;
plot(x,y,'*')
hold on
plot(x,Estimates(1).*log(Estimates(2).*x),'r')
```

Subroutine

```
function Error_Vector=nonlinmlf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(40,1).*40.+0.5);
r=o(1:5');
d=273+(r-1).*500;
% Defining parameters.
A=params(1)
B=params(2)
fc1=A.*log(B.*input);
% Calculating 'y' values for corresponding 'x' values.
fc2=A.*log(B.*d);
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if(fc2(i)==fc1(j))
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2)
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D6 Nonlinear Model – Gauss-Newton Method

Main Program

```
clear all;
clc;
% Define the data sets that you are trying to fit the
% function to
```

```
x=273:500:19773;
y=5.*log(55.*x)+2.*randn(size(x));

% Initialize the coefficients of the function
%X0=[1 1 1 1 1]';
%
%
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter','Largescale','on','LevenbergMarquardt
','off','maxFunEvals',400);
%
%
% Set an options file for LSQNONLIN to use the
% medium-scale algorithm
%options = optimset('Largescale','on');

% Calculate the new coefficients using LSQNONLIN
Estimates=lsqnonlin(@nonlinmlf,Starting,[],[],options,x,y);

% Plot the original and experimental data
clf;
plot(x,y,'*')
hold on
plot(x,Estimates(1).*log(Estimates(2).*x),'r')
```

Subroutine

```
function Error_Vector=nonlinmlf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(40,1).*40.+0.5);
r=o(1:5');
d=273+(r-1).*500;
% Defining parameters.
A=params(1)
B=params(2)
fc1=A.*log(B.*input);
% Calculating 'y' values for corresponding 'x' values.
fc2=A.*log(B.*d);
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if(fc2(i)==fc1(j))
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2)
% Total SSE
```

```
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D7 Multivariable Model – Nelder-Mead Simplex Method

Main Program

```
clear all;
clc;
% Generating random values of 'x' and 'y'.
x=0.01:0.5:10;
y=0.01:0.5:10;
z=0.5.*x.^0.5+2.*y.^0.5+randn(size(x));
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter');
% Optimizing parameters
Estimates=fminsearch(@multisimf,Starting,options,x,y)
% To check the fit
clf;
surf(x,y,z,'*')
hold on
surf(x,y,Estimates(1).*x.^0.5+Estimates(2).*y.^0.5)
```

Subroutine

```
function sse=multisimf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(20,1).*20.+0.5);
r=o(1:5');
d=0.01+(r-1).*0.5;
s=0.01+(r-1).*0.5;
% Defining parameters.
input
A=params(1)
B=params(2)
fc1=(A.*input.^0.5+B.*input.^0.5);
% Calculating 'y' values for corresponding 'x' values.
fc2=(A.*d.^0.5+B.*s.^0.5)';
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if fc2(i)==fc1(j)
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
```

```
ssd=sum(h(i).^2)
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D8 Multivariable Model - Marquardt-Levenberg Method

Main Program

```
clear all;
clc;
% Define the data sets that you are trying to fit the
% function to
x=0.01:0.5:10;
y=0.01:0.5:10;
z=0.5.*x.^0.5+2.*y.^0.5+3.*randn(size(x));
% Initialize the coefficients of the function
%X0=[1 1 1 1 1]';
%
%
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter','Largescale','off','LevenbergMarquard
t','on','maxFunEvals',400);
%
%
% Set an options file for LSQNONLIN to use the
% medium-scale algorithm
%options = optimset('Largescale','off');

% Calculate the new coefficients using LSQNONLIN
Estimates=lsqnonlin(@multimlf,Starting,[],[],options,x,y);

% Plot the original and experimental data
clf;
plot3(x,y,z,'*')
hold on
plot3(x,y,Estimates(1).*x.^0.5+Estimates(2).*y.^0.5)
```

Subroutine

```
function Error_Vector=multimlf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(20,1).*20.+0.5);
r=o(1:5');
d=0.01+(r-1).*0.5;
s=0.01+(r-1).*0.5;
% Defining parameters.
A=params(1)
B=params(2)
fc1=(A.*input.^0.5+B.*input.^0.5);
% Calculating 'y' values for corresponding 'x' values.
```

```
fc2=(A.*d.^0.5+B.*s.^0.5)';
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if fc2(i)==fc1(j)
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2)
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

D9 Multivariable Model - Gauss-Newton Method

Main Program

```
clear all;
clc;
% Define the data sets that you are trying to fit the
% function to
x=0.01:0.5:10;
y=0.01:0.5:10;
z=0.5.*x.^0.5+2.*y.^0.5+3.*randn(size(x));
% Initialize the coefficients of the function
%X0=[1 1 1 1 1]';
%
%
% Initial guess for the parameters.
Starting=rand(1,2);
options=optimset('Display','iter','Largescale','on','LevenbergMarquardt
','off','maxFunEvals',400);
%
%
% Set an options file for LSQNONLIN to use the
% medium-scale algorithm
%options = optimset('Largescale','on');

% Calculate the new coefficients using LSQNONLIN
Estimates=lsqnonlin(@multimlf,Starting,[],[],options,x,y);

% Plot the original and experimental data
clf;
plot3(x,y,z,'*')
hold on
plot3(x,y,Estimates(1).*x.^0.5+Estimates(2).*y.^0.5)
```

Subroutine

```
function Error_Vector=multimlf(params,input,Actual_Output)
% Selecting random 20 values of 'x'.
out_number = zeros(5,1);
o=round(rand(20,1).*20.+0.5);
r=o(1:5');
d=0.01+(r-1).*0.5;
s=0.01+(r-1).*0.5;
% Defining parameters.
A=params(1)
B=params(2)
fc1=(A.*input.^0.5+B.*input.^0.5);
% Calculating 'y' values for corresponding 'x' values.
fc2=(A.*d.^0.5+B.*s.^0.5)';
% Extracting the calculated fc2 values.
for i = 1:length(fc2)
    for j = 1:length(fc1)
        if fc2(i)==fc1(j)
            k(i) = j;
        end
    end
end

for i = 1:length(fc2)
    h(i) = fc2(i) - Actual_Output(k(i));
end
% SSD calculation.
ssd=sum(h(i).^2)
% Total SSE
Error_Vector=fc1 - Actual_Output;
sse=sum(Error_Vector.^2);
```

VITA

Venkatram Padmanabhan

Candidate for the Degree of

Master of Science

Thesis: A NOVEL STOPPING CRITERIONFOR OPTIMIZATION

Major Field: Chemical Engineering

Biographical:

Personal Data: Born in Coimbatore, Tamil Nadu, India, on November 5, 1980, to Dr. N.P.H. Padmanabhan and Mrs. Brinda Padmanabhan.

Education: Graduated from St. Patrick's High School, Hyderabad, India in May 1996; received Bachelor of Technology degree in Chemical Engineering from Chaitanya Bharathi Institute of Technology, India in June 2003. Completed the requirements for the Master of Science degree with a major in Chemical Engineering at Oklahoma State University in July 2005.

Experience: Summer internship at Bhabha Atomic Research Center, Mumbai, India, 2002; in-plant training at Dr. Reddy's Laboratories, Hyderabad, India, 2003; employed as Research Assistant by the School of Chemical Engineering, Oklahoma State University, August 2003 to August 2005; employed as Teaching Assistant at Oklahoma State University, August 2004 to August 2005.

Professional Memberships: Indian Institue of Chemical Engineers (IIChE), Instrumentation Systems and Automation Society (ISA).

Name: Venkatram Padmanabhan Date of Degree: July, 2005

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: A NOVEL STOPPING CRITERIONFOR OPTIMIZATION

Pages in Study: 132 Candidate for the Degree of Master of Science

Major Field: Chemical Engineering

Scope and Method of Study: A novel method for identification of steady state is demonstrated as the termination criterion for the optimization stage of modeling empirical data. The method was tested on a variety of applications. It is described, and its utility is demonstrated on modeling simulated data and is also validated using two laboratory scale experiments.

Findings and Conclusions: The novel stopping criterion for optimization, based on identifying steady state of a random subset of the sum of squared deviations with respect to iteration number, was formerly explored for neural network training. The novel stop-optimization criterion was tested on a different variety of applications involving various kinds of objective functions. On all the cases, the novel stop-optimization criterion gives equivalent results (as measured by model residuals) to the best possible results, with a sufficient (not excessive) number of iterations and without *a priori* knowledge of the optimization problem (scale, end-point values, and other classic stopping criteria).

ADVISER'S APPROVAL: Dr. R. Russell Rhinehart