A Comparison of Nonlinear

Regression Codes

by

Paul Fredrick Mondragon

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Mathematics with
Operations Research and Statistics Option.

New Mexico Institute of Mining and Technology
Socorro, New Mexico
May, 2003

ABSTRACT

In many instances during the process of data analysis, one faces a situation where at least one of the parameters of the model is interacting with the other parameters in a nonlinear manner. In such cases as this, one must use nonlinear regression in an effort to estimate the values of such parameters.

Nonlinear regression is inherently more difficult than linear regression. For this reason a software package is used. There are many software packages available which claim to be able to estimate such nonlinear parameters. With this in mind, one might well assume that the software packages are equal in ability.

In this study six readily available software packages, which are capable of solving such nonlinear regression problems are tested using standard test problems. The test problems vary in level of difficulty, as well as number of parameters and number of observations. The software packages are then compared based upon accuracy, robustness, features, and ease of use.

Table of Contents

I. Introduction:

The goal of this study is to compare the nonlinear regression capabilities of several software packages using the nonlinear regression data sets available from the National Institute of Standards and Technology (NIST) Statistical Reference Datasets (StRD) website located at http://www.itl.nist.gov/div898/strd/ . The software packages considered in this study are:

1. MATLAB codes from Hans Bruun Nielsen
2. Gaussfit
3. Gnuplot
4. SAS
5. Microsoft Excel
6. Minpack

The Statistical Reference Datasets Project was developed by the Statistical Engineering Division and the Computational Sciences Division within the Information Technology Laboratory of the National Institute of Standards and Technology. There are four areas covered by the StRD: univariate summary statistics, one–way analysis of variance, linear regression, and nonlinear regression. Each area includes problems of lower, average, and higher difficulty. The difficulty level is determined by the sources of inaccuracy: truncation error, cancellation error, and accumulation error. Truncation error relates to the inexact binary representation error in storing decimal numbers. Cancellation error results from the "stiffness" i.e. the number of constant leading digits in the data sets. Since the total number of arithmetic computations is proportional to the size of a data set, the accumulation error may increase as the number of observations increase due to the accumulation of small errors. [8]

The nonlinear regression problems were solved by the NIST using quadruple precision (128 bits) and two public domain programs with different algorithms and different implementations; the convergence criterion was residual sum of squares (RSS) and the tolerance was 1E-36. Certified values were obtained by rounding the final solutions to 11 significant digits. Each of the two public domain programs, using only double precision, could achieve 10 digits of accuracy for every problem. [11]

I. A. Comparison Criteria

As is the case in any comparison study there must be criteria upon which the comparison is based. The criteria we will use in the evaluation of these software packages are accuracy, robustness, ease of use, and features.

Accuracy will be determined using the log relative error (LRE) formula,

$$\lambda_q = -\log_{10}\left[\frac{|q-c|}{|c|}\right]. \qquad (1)$$

where q is the value of the parameter estimated by the code being tested and c is the certified value. In the event that $q = c$ exactly then λ_q is not defined, in which case it should be set equal to the number of digits in c. It is possible for an LRE to exceed the number of digits in c; for example, it is possible to calculate an LRE of 11.4 even though c contains only 11 digits. In part, this is because a double precision computer will "pad" the 11 digits with zeros. In such a case, λ_q should be set equal to the number of digits in c. Finally, any λ_q less than unity should be set to zero. [11]

Robustness is an important characteristic for a software package. In terms of accuracy, we are concerned with each specific problem as individuals. Robustness, however, is a measure of how the software packages performed on the problems as a set. In other words, we want to have a sense of how reliable the software package is so that we might have some level of confidence that it will solve a particular nonlinear regression problem other than those listed in the NIST StRD. In this sense robustness may very well be more important to the user than accuracy. Certainly the user would want parameter estimates to be accurate to some level, but accuracy to 11 digits is often not particularly useful in practical application. However, the user would want to be confident that the software package they are using will generate parameter estimates accurate to perhaps 4 or 5 digits on most any problem they attempt to solve. If, on the other hand, a software package is extremely accurate on some problems, but returns a solution which is not anywhere close to actual values on other problems, certainly the user would want to use this software package with extreme caution.

Comparing the features of these various software packages is a rather difficult comparison to make. It is difficult because some of these software packages are designed simply for nonlinear regression problems, whereas some of them are designed for many more uses. The packages that are designed for more than simply nonlinear regression problems will have more features, but perhaps not more features concerning the linear regression problems. It is these features which we are interested in for this study.

Admittedly a criterion such as ease of use is somewhat subjective, as each user has their own preferences. However, it will undoubtedly be the case that some of the software packages will be easier to use than others. Such information might well be useful for consideration in determining which package to use for the solving of nonlinear regression problems. Such things as amount of programming necessary, data input, and whether the package writes over files will be considered.

I. B. Literature Search

In their paper, "Assessing the Reliability of Web-Based Statistical Software",[8] A. M. Kitchen, R. Drachenberg, and J. Symanzik used the statistical datasets from the National Institute of Standards and Technology to evaluate the accuracy and precision of WebStat 2.0 (found at http://www.stat.sc.edu/webstat/) and Statlets (found at http://www.statlets.com/statletsindex.htm). In this paper, the two software packages were assessed by comparing the results of univariate summary statistics, analysis of variance, and linear regression operations with the certified values given in the StRD. Nonlinear

regression analysis was not evaluated because these packages were unable to carry out nonlinear regression procedures. [8]

In addition, K. L. Hiebert has written a paper, "An Evaluation of Mathematical Software That Solves Nonlinear Least Squares Problems" [5]. In this paper, Hiebert compares 12 FORTRAN codes on 36 separate problems. Twenty-eight of the problems used by Hiebert are given by Dennis, J.E., Gay, D.M. and Welch, R.E. [3], with the other eight problems given by More, J.J, Garbow, B.S., and Hillstrom, K.E. [12].

In their paper, "Testing Unconstrained Optimization Software" [12], More, J.J., Garbow, B.S., and Hillstrom, K.E., used FORTRAN subroutines to test 35 problems. These 35 problems were a mixture of systems of nonlinear equations, nonlinear least – squares, and unconstrained minimization.

In this study we will evaluate the ability of the six software packages mentioned to accurately estimate the parameters for the nonlinear regression problems provided by the National Institute of Standards and Technology Statistical Reference Datasets.

II. Nonlinear Least Squares

In many areas of the physical, chemical, biological, and engineering sciences, at least one of the parameters enters the model in a nonlinear manner. The development of the least squares estimators for a nonlinear model brings about complications not encountered in the case of a linear model. Let us consider a nonlinear model of the general form

$$y_i = g(x_i;\beta) + \varepsilon_i \qquad (i = 1,2,\cdots, m) \tag{2}$$

where β is a vector containing n parameters and $m > n$. We assume further that f is nonlinear in $\beta^T = [\beta_1, \beta_2, \cdots, \beta_n]$.

A popular method for estimating the unknown parameters in a nonlinear regression function is the method of least squares. According to this method, the estimates of $\beta_1, \beta_2, \cdots, \beta_n$ are obtained by minimizing the quantity

$$\sum_{i=1}^{m} f_i(\beta)^2 , \tag{3}$$

the sum of the squares of the errors of the predictions, where

$$f_i(\beta) = y_i - g(x_i;\beta) . \tag{4}$$

III. Nonlinear Regression

Since we are assuming that the errors are random, we are interested in their underlying probability distribution. We generally assume that the errors are normally distributed with mean 0 and variance σ^2. This is a reasonable assumption if the data are measurements of some type. More importantly than this, however, is that if the errors are normally distributed, the maximum likelihood estimate of β is the estimate obtained using the least squares method as we will now show.

The maximum likelihood estimate of β is obtained by maximizing the likelihood function. If the ε_i's are independently and identically distributed (i.i.d.) with density function $\sigma^{-1}h(\varepsilon/\sigma)$ so that h is the error distribution for the errors standardized to have unit variance, then the likelihood function is

$$p(y \mid \beta, \sigma^2) = \prod_{i=1}^{m} \left[\sigma^{-1} h\left(\frac{y_i - g(x_i;\beta)}{\sigma} \right)^2 \right]. \tag{5}$$

Since the ε_i's are i.i.d. $N(0, \sigma^2)$, (5) becomes

$$p(y \mid \beta, \sigma^2) = (2\pi\sigma^2)^{-\frac{m}{2}} \exp\left(-\frac{1}{2} \sum_{i=1}^{m} \frac{[y_i - g(x_i;\beta)]^2}{\sigma^2} \right). \tag{6}$$

Ignoring constants, we denote the logarithm of the above likelihood by $L(\beta, \sigma^2)$ and obtain

$$L(\beta, \sigma^2) = -\frac{m}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{m} [y_i - g(x_i;\beta)]^2$$

$$= -\frac{m}{2} \log \sigma^2 - \frac{1}{2\sigma^2} S(\beta) \tag{7}$$

where,

$$S(\beta) = \sum_{i=1}^{m} [y_i - g(x_i;\beta)]^2 .$$

Given σ^2, (7) is maximized with respect to β when $S(\beta)$ is minimized, that is when $\beta = \hat{\beta}$ (the least squares estimate). Furthermore, $\partial L / \partial \sigma^2 = 0$ has solution $\sigma^2 = S(\beta)/m$, which gives a maximum (for given β) as the second derivative is negative. This suggests

that $\hat{\beta}$ and $\hat{\sigma}^2 = S(\hat{\beta})/m$ are the maximum likelihood estimates as we now verify directly. Since $S(\beta) \geq S(\hat{\beta})$

$$L\left(\hat{\beta},\hat{\sigma}^2\right) - L\left(\beta,\sigma^2\right) = -\frac{m}{2}\log\hat{\sigma}^2 - \frac{m}{2} - L\left(\beta,\sigma^2\right)$$

$$\geq -\frac{m}{2}\log\frac{\hat{\sigma}^2}{\sigma^2} - \frac{m}{2} + \frac{1}{2}\frac{S(\hat{\beta})}{\sigma^2}$$

$$= -\frac{m}{2}\left(\log\frac{\hat{\sigma}^2}{\sigma^2} + 1 - \frac{\hat{\sigma}^2}{\sigma^2}\right)$$

$$\geq 0\ ,$$

as $\log x \leq x - 1$ for $x \geq 0$. The m.l.e. of $\hat{\beta}$ and $\hat{\sigma}^2$ are substituted into equation (6) and we obtain

$$p(y \mid \hat{\beta},\hat{\sigma}^2) = (2\pi\hat{\sigma}^2)^{-m/2}\exp\left(-\frac{m}{2}\right). \tag{8}$$

Once we have estimated the parameters using the least square method, we can determine some characteristics regarding the estimates. The fitted parameters, $\hat{\beta}$, are approximately normally distributed with mean $\hat{\beta}$ and variance-covariance matrix given by

$$\Sigma = \left(J(\hat{\beta})^T J(\hat{\beta})\right)^{-1}. \tag{9}$$

Where $J(\hat{\beta})$ is the Jacobian of

$$f_i(\hat{\beta}) = \frac{y_i - g(x_i;\hat{\beta})}{\sigma_i}. \tag{10}$$

The (1-α)100% confidence interval for $\hat{\beta}_i$ is given by

$$\hat{\beta}_i \pm Z_{(1-\alpha)/2}\sqrt{\Sigma_{ii}}\,, \tag{11}$$

where $Z_{(1-\alpha)/2}$ is obtained from the standard normal table for some α.

If we know the assumption that the error terms are normally distributed with mean 0 and variance σ^2 does not hold, then we can manipulate the data so that they are reasonable. If, for example, the errors do not have the same variance but rather, each ε_i has a variance σ_i^2, then instead of solving the problem given by

$$\min \sum_{i=1}^{m} \left(y_i - g(x_i;\beta) \right)^2, \tag{12}$$

we solve the problem given by

$$\min \sum_{i=1}^{m} \left(\frac{y_i - g(x_i;\beta)}{\sigma_i} \right)^2. \tag{13}$$

If the σ_i's are unknown, then we might assume that all the σ_i's are equal and minimize (12). We can then estimate this value for the variance as

$$s = \sqrt{\frac{\sum_{i=1}^{m} \left(y_i - g(x_i;\beta) \right)^2}{m-n}} \tag{14}$$

where m is the number of observations in the model and n is the number of parameters estimated in the model. Here the variance-covariance matrix is given by

$$\Sigma = s^2 \left(J(\hat{\beta})^T J(\hat{\beta}) \right)^{-1}. \tag{15}$$

The (1-α)100% confidence interval for $\hat{\beta}_i$ is given by

$$\hat{\beta}_i \pm t_{m-n,(1-\alpha)/2}\sqrt{\Sigma_{ii}}\,, \tag{16}$$

where $t_{m-n,(1-\alpha)/2}$ is obtained from a t distribution table.

III. A. Why we Use Nonlinear Regression

One of the major advantages in using nonlinear regression is the broad range of functions that can be fit. Many scientific or physical processes are inherently nonlinear; therefore to attempt a linear fit of such a model would necessarily cause poor results. For example, studies in population quite often follow exponential pattern, which cannot be modeled in a linear fashion.

Additionally, we can take advantage of the fairly well developed theory for computing confidence, prediction, and calibration intervals. It is true that quite often the probabilistic interpretation of the intervals are only approximately correct, but these intervals still work very well in practice.

Nonlinear regression also shares the characteristic with linear regression of having very efficient use of the data. That is, nonlinear regression can give good estimates of the unknown parameters in the model using relatively small data sets. [4]

III. B. The Gauss-Newton Method

Let us consider the Taylor series expansion with the remainder term as follows:

$$f(\hat{\beta}+p) = f(\hat{\beta}) + \nabla f(\hat{\beta})^T p + \frac{1}{2} p^T \nabla^2 f(\xi) p \qquad (17)$$

where p is a nonzero vector and ξ is a point between $\hat{\beta}$ and $\hat{\beta}+p$. If $\hat{\beta}$ is a local minimizer, there can be no feasible descent direction at $\hat{\beta}$. Hence,

$$\nabla f(\hat{\beta})^T p \geq 0 \text{ for all feasible directions } p.$$

Since a least–squares problem is an unconstrained problem, all directions are feasible, therefore the gradient at β_* must be zero. Thus if β_* is a local minimizer of f, then

$$\nabla f(\hat{\beta}) = 0.$$

A point satisfying this condition is a stationary point of the function f.

In order to distinguish whether a stationary point is a local minimizer, a local maximizer, or a saddle point (a stationary point that is neither a minimizer or maximizer), one must consider the second derivatives. Consider again the Taylor series expansion at $\beta = \hat{\beta}+p$, but now using the result that $\nabla f(\hat{\beta}) = 0$.

$$f(\beta) = f(\hat{\beta}+p) = f(\hat{\beta}) + \frac{1}{2} p^T \nabla^2 f(\xi) p \qquad (18)$$

If $\nabla^2 f(\hat{\beta})$ is not positive semi-definite then for some $\upsilon, \upsilon^T \nabla^2 f(\hat{\beta})\upsilon < 0$. Then it is also true that $\upsilon^T \nabla^2 f(\xi)\upsilon < 0$ if $\left\| \xi - \hat{\beta} \right\|$ is small, since $\nabla^2 f$ is assumed to be continuous at $\hat{\beta}$. If p is chosen as some sufficiently small multiple of υ, then the point ξ will be close enough to $\hat{\beta}$ to guarantee (via the Taylor series) that $f(\beta) < f(\hat{\beta})$ which is a contradiction. Therefore, $\nabla^2 f(\hat{\beta})$ must be positive semi-definite for $\hat{\beta}$ to be a local minimizer.

Newton's method is an algorithm for finding a zero of a nonlinear function. To use Newton's method for optimization, we apply it to the system of equations given by $\nabla f(\hat{\beta}) = 0$. Since the Jacobian of $\nabla f(\beta)$ is $\nabla^2 f(\beta)$, this leads to the formula

$$\beta_{k+1} = \beta_k - [\nabla^2 f(\beta_k)]^{-1} \nabla f(\beta_k). \tag{19}$$

Newton's method is often written as $\beta_{k+1} = \beta_k + p_k$ where p_k is the solution to the Newton equations:

$$[\nabla^2 f(\beta_k)]p = -\nabla f(\beta_k). \tag{20}$$

This emphasizes that the step p_k is usually obtained by solving a linear system of equations rather than by computing the inverse of the Hessian.

Newton's method has been derived by finding a linear approximation to a nonlinear function via the Taylor series. The formula for Newton's method represents a step to a zero of this linear approximation. For the nonlinear equation $\nabla f(\beta) = 0$ this linear approximation is

$$\nabla f(\beta_k + p) \approx \nabla f(\beta_k) + \nabla^2 f(\beta_k) p. \tag{21}$$

The linear approximation is the gradient of the quadratic function

$$Q(p) \equiv f(\beta_k) + \nabla f(\beta_k)^T p + \frac{1}{2} p^T \nabla^2 f(\beta_k) p. \tag{22}$$

Q(p) corresponds to the first three terms of a Taylor series expansion for *f* about the point β_*.

The quadratic function *Q* provides a new interpretation of Newton's method for minimizing *f*. At every iteration Newton's method approximates $f(\beta)$ by *Q*(*p*), the first three terms of its Taylor series about the point β_k; minimizes *Q* as a function of *p*; and then sets $\beta_{k+1} = \beta_k + p$. Hence at each iteration we are approximating the nonlinear function by a quadratic model. [13]

The method of nonlinear least–squares data fitting, as given by nonlinear regression, also has a special form for the gradient and Hessian. Let us express the problem as

$$\min f(\beta) = \frac{1}{2}\sum_{i=1}^{m} f_i(\beta)^2 = \frac{1}{2} F(\beta)^T F(\beta) \tag{23}$$

where F is the vector–valued function

$$F(\beta) = (f_1(\beta), f_2(\beta), \cdots, f_m(\beta))^T \tag{24}$$

Note that the scaling by $\frac{1}{2}$ is to make the derivatives less cluttered. The components of $\nabla f(\beta)$ can be derived as follows:

$$\nabla f(\beta) = J(\beta)^T F(\beta)\,. \tag{25}$$

where J is the Jacobian matrix with ijth element

$$J_{ij} = \frac{\partial g(x_i;\beta)}{\partial \beta_j},\ i = 1,2,\cdots,m\text{, and } j = 1,2,\cdots,n\,. \tag{26}$$

$\nabla^2 f(\beta)$ can be derived by differentiating this formula with respect to the β_j's:

$$\nabla^2 f(\beta) = J(\beta)^T J(\beta) + \sum_{i=1}^{m} f_i(\beta)\nabla^2 f_i(\beta)\,. \tag{27}$$

Since we expect $f_i(\hat{\beta})$ to be approximately zero, the summation term can be ignored as $\beta \to \hat{\beta}$. Therefore, we can approximate $\nabla^2 f(\beta)$ as

$$\nabla^2 f(\beta) = J(\beta)^T J(\beta)\,, \tag{28}$$

For the Gauss – Newton method we can use this approximation for the Hessian and solve

$$J(\beta_k)^T J(\beta_k) p_k = -J(\beta_k)^T F(\beta_k) \tag{29}$$

to calculate p_k. Then let $\beta_{k+1} = \beta_k + p_k$.

III. C. The Levenberg – Marquardt Method

As might be expected, Newton's method has a quadratic rate of convergence except in "degenerate" cases; it can sometimes diverge or fail. For example, if $J(\beta)^T J(\beta)$ is singular, then (29) will have no unique solution. As a result, some modifications to Newton's method have been made. We will now look at one such modification – the Levenberg – Marquardt method. [13]

The method which is now know as the Levenberg - Marquardt method is a result of the work of Donald Marquardt.[9] The insights leading to this method arose from Marquardt's experience with several two – parameter estimation problems. The intuition of Marquardt's chemical engineering colleagues was often sufficient to provide good starting parameters for methods such as steepest descent, which iterates to the best estimates of the parameters by heading straight down the wall of a valley in the cost function surface. With poor initial guesses, however, the methods take many iterations to converge, or may not converge at all.

As Marquardt began to plot the contours of the cost functions, he began to observe a generic geometric problem. Methods such as steepest descent, which follow the gradient down the function surface move in a direction that is nearly orthogonal to the Taylor series methods which linearized the cost function. This geometric conflict was a consequence of the long, narrow valleys in the cost function. The ideal method would find an angle of descent intermediate between these two extremes. At the same time, the step size would require adjustment to prevent stepping across the valley and entirely missing the floor where the best parameter values lay.

Marquardt recognized that these goals could be accomplished by adding a properly sized parameter to the diagonal of the system of equations defining the iterates. Marquardt did not require finding a local minimum of the cost function at each step. This avoided the slow convergence often encountered by the steepest descent method as it travels along the narrow path crossing many times the valley in the cost function floor. As a result of these modifications to Levenberg's prior work, the Levenberg – Marquardt method has proved to be an effective and popular way to solve nonlinear least squares problems. [2]

III. C. 1. How the Levenberg – Marquardt Method Works

Rather than approximating the Hessian as in (28), Marquardt noted that the summation term in (27) can be approximated by λI where $\lambda \geq 0$. Using this we approximate the Hessian as

$$\nabla^2 f(\beta_k) \approx J(\beta_k)^T J(\beta_k) + \lambda I . \tag{30}$$

Now to find our search direction, *p*, we solve the equation

$$[J(\beta)^T J(\beta) + \lambda I]p = -J(\beta)^T F(\beta) . \tag{31}$$

After finding p, we evaluate $f(\beta + p)$. If there has been an improvement in our function value then we let $\beta = \beta + p$ and $\lambda = \frac{\lambda}{2}$. We then check our termination criteria. If the termination criteria are not met, then we proceed with another iteration. If, however, the evaluation of $f(\beta + p)$ does not give us an improvement in our function value, then we let $\lambda = 2\lambda$ but do not change β. We then solve the above system of equations for p and once again check $f(\beta + p)$.

In essence, the Levenberg – Marquardt method takes on a search direction similar to the steepest descent method when β is "far" from $\hat{\beta}$. However, as β gets closer to $\hat{\beta}$, it takes on a search direction like that of Newton's method. [4]

III. C. 2. Why We Use the Levenberg – Marquardt Method

As stated above the Levenberg – Marquardt method has proved to be a popular tool for solving nonlinear least squares problems. The reasons for this popularity are essentially threefold.

First of all, there is a consideration of storage space. Like other methods for nonlinear least squares problems there is an approximation to the Hessian matrix. Since we are able to approximate the Hessian matrix we do not have to store this *nxn* matrix. The particular approximation of the Hessian used by the Levenberg – Marquardt method is given by λI, where $\lambda \geq 0$. The benefit of this is that the search direction is calculated by solving the linear system given in (31). Note that this system only involves the original vector function and the gradient.

The second reason is an issue of computational effectiveness. If on an intermediate step it is determined that there has not been an improvement in the value of the cost function, then it is not necessary to recalculate the gradient at that point. Rather we would simply adjust λ and continue through the algorithm. In addition to this, we have no need for calculating the Hessian, which may be very costly to calculate.

Thirdly, the Levenberg – Marquardt method allows for movement along the contour of the cost function, which allows the algorithm to solve the problem quicker than many other algorithms.

The combination of storage space, computational effectiveness, and iteration efficiency make the Levenberg – Marquardt a powerful tool for the nonlinear least squares problems that are so common in the engineering and scientific disciplines.

IV. The Software Packages

We will now take a closer look at the various software packages that we have chosen for this comparative study. Some of the packages are simply parts of a larger package, such as Microsoft Excel and SAS. In this case, we will simply consider the parts of the larger package which were used in the completion of this study. Others in the set of packages used are designed exclusively for solving nonlinear least – squares problems.

IV. A. HBN Matlab Code

The first software package used in this study is the Matlab code written by Hans Bruun Nielson. Nielsen has written several codes to solve nonlinear least – squares problems. Using the Levenberg - Marquardt method, he has written code for which the Jacobian is approximated. Additionally he has written code for which the Jacobian is calculated analytically. For the purpose of this study the Jacobian was calculated analytically.

The Marquardt program has four input arguments. The first input argument is a function that calculates the function values, $f_i(x)\ \ i = 1,2,\ldots,m$ and the Jacobian matrix. While combining the calculations of the function values and Jacobian matrix into one function reduces the number of input parameters, it also causes the program to perform some unnecessary calculations. For example, if after taking a step it is determined that the functional value has not improved, calculating the Jacobian at the new point is not necessary. Rather one ought to adjust the damping parameter, λ , and calculate a new x. When the calculations of the function values and the Jacobian are combined in this manner, the Jacobian will automatically be calculated even in those instances when it is not necessary.

The second of the input arguments is for an array with the coordinates of the data points, or it may be a dummy variable. For this study I kept this variable as a dummy variable.

The third input argument is the starting guess for the values of the parameters. For each problem, this would be a column vector with p elements. There are also two starting values for the parameters, so the program was run twice for each problem.

The final input argument is a four - element vector used as parameters for the program. The first element is used as the starting value for the damping parameter, λ . The rest of the elements are used as stopping criteria for the program. The program terminates if either $\left\|F^T\right\|_\infty \le$ opts(2), or $\left\|\Delta x\right\|_2 \le$ opts(3), or if the number of iterations exceeds opts(4).

In addition to the four input arguments, Nielsen's Marquardt code has three output arguments. The first is a vector consisting of the optimal value for the parameters found by the code. By setting the Matlab format to "long e" I was able to output more than 11 significant digits in order to properly compare the optimal values returned to the certified values.

The second output argument is a six element vector which gives data regarding the performance of the code. The first four elements are the final values of $F(x)$, $\left\|F^T\right\|_\infty$,

$\|\Delta x\|_2$, and $\lambda / \max(A(i,i))$. The fifth element of this vector is the number of iterations the code performed before stopping. The sixth element reports which of the termination criteria was active to terminate the code. If this sixth element is a "1", then the code terminated due to a small gradient. If it is a "2", then the code terminated due to a small change in the parameter values. If it is a "3", then the code terminated due to the number of iterations exceeding the maximum. If this sixth element is a "4", then the code has calculated a singular matrix. The user is then to restart with the current parameter estimates as the initial guess with a larger value for λ.

Nielsen also included an auxiliary function which checks the function call statement to ensure that the input arguments are given as expected by the main program. For example, in this function he checks to make sure that the initial guess is given as a vector. He also checks that the sizes for the vector which gives the function values and the Jacobian matrix are compatible.

IV. B. GaussFit

In this study version 3.53 of Gaussfit was used using Linux as the operating system.

GaussFit was designed for astrometric data reduction with data from the NASA Hubble Space Telescope. It was designed to be a flexible least squares package so that astrometric models could easily and quickly be written, tested and modified.

A unique feature of GaussFit is that although it is a special purpose system designed for estimation problems, it includes a full-featured programming language which has all the power of traditional languages such as C, Pascal, and FORTRAN. This language possesses a complete set of looping and conditional statements as well as a modern nested statement structure. Variables and arrays may be freely created and used by the programmer. There is therefore no theoretical limit to the complexity of model that can be expressed in the GaussFit programming language. [6]

One of the onerous tasks that faces the implementer of a least squares problem, particularly if the problem is nonlinear, is the calculation of the partial derivatives with respect to the parameters and observations that are required in order to form the equations of condition and the constraint equations. GaussFit solves this problem automatically using a built-in algebraic manipulator to calculate all of the required partial derivatives. Every expression that the user's model computes will carry all of the required derivative information along with it.

This is very convenient, especially when a condition or a constraint equation involves a complex calculation. For example, if the calculation can only be expressed algorithmically, GaussFit will automatically carry along all derivative information at each step of the calculation. The derivatives are calculated analytically at each step. No numerical approximations are used. [6]

In order to run GaussFit for the estimation of the unknown parameters in a nonlinear least squares problem the user must provide four files. These four files are:

1. model file
2. parameter file
3. data file

4. environment file.

The model file is made up of three sections. In the first section the names of the parameters that are used in the function and are being estimated. The second section involves the data. The observation statement declares that the data associated with that variable contains errors. If no errors are to be assumed for the values of the data associated with a variable, then the data command is used. In this study there are assumed to be errors in the 'y' data due to the inherent errors in the model, while the 'x' data are assumed to be without errors. The third section of the model file then is the main program. It is in this section that the function is defined and calculated. In this study, a while loop was used to calculate the function values for each of the pairs of data. The model file, unlike the parameter and environment and data files, is not changed after running the GaussFit program.

The parameter file defines the parameters being estimated. The first line names the parameters, while the second line declares the type of variable the parameters are. In this study, all of the parameters were declared to be double precision. The third line of the parameter file states the initial values for the parameters. Running the GaussFit program will change the parameter file. GaussFit will write into the parameter file the change in each parameter during the last performed iteration. It will also write the calculated standard deviation for the parameters.

The data file used by GaussFit has one column for each observed quantity. The first line of each column is the variable for which that data is assigned. The variables are then declared as to their type. Again, in this study all data was declared to be double precision. Weights and variances may also be included in the data file. After running the GaussFit program another column will be added to the data file. This column will contain the residuals calculated by the program for the most recent iteration. The actual observation values will not be changed.

The environment file contains information required for the reduction. This file is composed of setting certain keywords to appropriate values in order that the program will perform as desired. I will briefly describe the keywords used in this study. The keywords **results**, **data**, and **Params**, give the names of the files which are to be used by the program to output the results, contain the data, and find the definitions of the parameters respectively. The keyword **tol** gives the value of the relative tolerance (e.g., the maximum change in computed values from one iteration to the next). The maximum number of iterations is set using the **iters** keyword. The **prmat** keyword specifies that certain intermediate matrices will be printed. In this case **prmat** was set to 0 in order that none of the intermediate matrices were printed. The **prvar** keyword controls the printing of the variance – covariance matrix. The keywords **lambda** and **factor** are specific to using the Lenvenberg – Marquardt method. The **lambda** keyword sets the initial damping factor, while the **factor** keyword sets the value whereby the damping factor is reduced. After running the GaussFit program the current scale factor and current sigma value will be added into the environment file. [6]

IV. C. Gnuplot

Gnuplot is a command-driven interactive function plotting program capable of a variety of tasks. Included among these tasks are plotting both two- or three-dimensional functions in a variety of formats, computations in integer, floating point, and complex arithmetic, and support for a variety of operating systems. [1]

For this study gnuplot version 3.7 patchlevel 3 was used. Initially gnuplot displayed only approximately 6 digits in its solutions to the estimation of the parameters. I e-mailed the technical service mailing list and was essentially told that the additional digits were not statistically significant. While this is certainly true, by the nature of this study it is essential that we have at least 11 significant digits for the estimation in order that we may compare the estimates to the certified values provided in the NIST StRD datasets. Dr. Brian Borchers edited the underlying code changing the display command from "%g" to "%.20e" which allowed the program to display 20 digits.

The "fit" command can fit a user-defined function to a set of data points (x,y) or (x,y,z), using an implementation of the nonlinear least-squares Marquardt – Levenberg algorithm. Any user-defined variable occurring in the function body may serve as a fit parameter, but the return type of the function must be real.

After each iteration step, detailed information about the current state of the fit is written to the display. The same information about the initial and final states is written to a log file, "fit.log". This file is always appended to, so as to not lose any previous fit history. After each problem I renamed the .log file in order to have quick access to the results for each individual problem.

Adjustable parameters can be specified by a comma separated list of variable names using the **via** keyword. Any variable that is not already defined will be created with an initial value of 1.0. However it is best to define the variables before the fit command with appropriate starting values.

Rather than determine confidence intervals, fit reports parameter error estimates which are readily obtained from the variance-covariance matrix after the final iteration. By convention, these estimates are called "standard errors" or "asymptotic standard errors", since they are calculated in the same way as the standard errors (standard deviation of each parameter) of a linear least squares problem, even though the statistical conditions for designating the quantity calculated to be a standard deviation are not generally valid for the nonlinear least squares problem. The asymptotic standard errors are generally over-optimistic and should not used for determining confidence intervals, but are useful for qualitative purposes.

The final solution also produces a correlation matrix, which gives an indication of the correlation of parameters in the region of the solution. The main diagonal elements, autocorrelations, are all 1.0; if all parameters were independent, all other elements would be nearly zero. Two variables which completely compensate each other would have an off-diagonal element of unit magnitude, with a sign depending on whether the relation is proportional or inversely proportional. The smaller the magnitudes of the off-diagonal elements, the closer the estimates of the standard deviation of each parameter would be to the asymptotic standard error. [1]

The user has the ability to set various program parameters with gnuplot. The command "FIT_LIMIT" will set the tolerance for the program. The default tolerance is

set equal to 1.0e-5. When the sum of the squared residuals changes between two iteration steps by a factor less than this tolerance the fit is considered to have converged. Additionally the maximum number of iterations may be set by the command "FIT_MAXITER". The default value is set at no limit to the number of iterations. The initial values for the Levenberg-Marquardt damping parameter, λ, as well as the factor by which λ is increased or decreased can be set by the user with the commands "FIT_START_LAMBDA" and "FIT_LAMBDA_FACTOR" respectively. The default initial value of λ is calculated within the program, while the default initial value for the λ - factor is set to 10.0. For the purposes of this study FIT_LIMIT was set to 1.0e-15, with the default values for the other program parameters. [1]

IV. D. SAS

For this study SAS Release 8.02.02 for use with Linux was used.

SAS is a multi-purpose commercial statistical software package. It is far beyond the scope of this paper to cover the many procedures found within the SAS package. Instead, we will concern ourselves only with the NLIN procedure which computes least squares or weighted least squares estimates of the parameters of a nonlinear model. Additionally, we will only consider the program parameters which were used in this study and direct the reader to the listed resources for additional information.

PROC NLIN fits nonlinear regression models using the least squares method. The user must supply the regression expression, declare parameter names, supply starting values for the parameters, and supply derivatives of the model with respect to the parameters. The NLIN procedure is capable of use any of the following five iterative methods:

- Steepest descent or gradient method
- Newton method
- Modified Gauss-Newton method
- Levenberg-Marquardt method
- Multivariate secant or false position (DUD) method

The Gauss-Newton and Levenberg-Marquardt iterative methods regress the residuals unto partial derivatives of the model with respect to the parameters until the estimates converge or the set maximum number of iterations is achieved. The Newton iterative method regresses the residuals onto a function of the first and second partial derivatives of the model with respect to the parameters until the estimates converge or the set maximum number of iterations is achieved.

After declaring the data (either listing data in the program itself or giving a file where the data is located), the **proc nlin** statement invokes the procedure. Several options are available for use in this statement. The "DATA" statement names the SAS data set containing the data to be analyzed by PROC NLIN. The "METHOD" statement specifies the iterative method NLIN uses. For this study METHOD=Marquardt. The "MAXITER" option sets the maximum number of iterations performed by PROC NLIN. For this study MAXITER was set equal to 500.

The "CONVERGE" statement uses the change in the LOSS function as the convergence criterion. The iterations are said to have converged for CONVERGE=c if

$$\frac{\left(LOSS_{i-1} - LOSS_i\right)}{\left(LOSS_i + 10^{-6}\right)} < c$$

where $LOSS_i$ is the *LOSS* for the *i*th iteration. The default *LOSS* function is the sum of squared errors (SSE). For this study CONVERGE was set equal to 1.0e-15.

The **model** statement defines the prediction equation by declaring the dependent variable and defining an expression that evaluates predicted values. The expression can be any valid SAS expression yielding a numerical result. The expression can include parameter names, variables in the data set, and variables created by program statements in the NLIN procedure. A model statement must appear in the NLIN procedure.

SAS requires the partial derivatives of the parameters to be estimated (with the exception being when using the secant method). The derivatives are stated using a statement such as "der.b0=<expression>". This statement defines the first partial derivative of the function with respect to the parameter b0. This derivative now can be used as a variable in statements which follow.

The data set produced by the OUTEST option in the PROC NLIN statement contains the parameter estimates for each iteration. The variable _ITER_ contains the iteration number. The variable _TYPE_ denotes whether the observation contains iteration parameter estimates (**ITER**), final parameter estimates (**FINAL**), or covariance estimates (**COVB**). The variable _NAME_ contains the parameter name for covariances, and the variable _SSE_ contains the objective function value for the parameter estimates.

In addition to the output data sets, NLIN also produces the estimates of the parameters and the residual Sums of Squares determined at each iteration. If the convergence criterion is met, NLIN prints an analysis of variance table including as sources of variation Regression, Residual, Uncorrected Total, and Corrected Total. Also printed are an asymptotically valid standard error of the estimate, an asymptotic 95% Confidence Interval for the estimate of the parameter, and an asymptotic Correlation Matrix of the parameters.

The default settings for SAS to output the estimates of the parameters display approximately six significant digits. Again, for comparison purposes it is desirable to have at least 11 significant digits. By running a print procedure I was able to format the output in a manner consistent with the purpose of this study. [14]

IV. E. Microsoft Excel

Just as is the case with SAS, Microsoft Excel 2000 is a multi-purpose software package. As only a small part of its capabilities were used during the process of this study, we will limit our discussion of Excel to its **Solver** capabilities.

The Excel Solver function is a self-contained function in that all of the data must be located somewhere on the spreadsheet. The Solver allows the user to find a solution to a function that contains up to 200 variables and up to 100 constraints on those variables.

For the Solver function to be configured properly the user must lay out the spreadsheet. In addition to the data and the function to be minimized the starting values for the parameters must be located on the worksheet. A target cell contains the value of the function to be minimized. When the Solver function is selected from the Tools menu, a dialog box will appear. In this dialog box the user is given the opportunity to set the function to a specific value, minimize the function, or maximize the function. The user must place the location of the target cell. Additionally, the user can list any constraints upon the parameters.

Within the options menu the user can adjust the parameters of the Solver program. The user can set the maximum amount of time or the maximum number of iterations that Solver will take to solve the problem. The user can also set the precision level of the solutions and the tolerance level for the program. The option to see results after each iteration is given. Solver calculates numerical approximations to the derivatives based upon either forward or central derivatives depending on the choice of the user. The user also chooses between a Quasi-Newton or a Conjugate Gradient method. Automatic scaling is yet another option for the user to select.

Solver generates three reports at the conclusion of the program. The Answer Report gives the original values for the target cell as well as the adjustable cells (parameters). Listed also are the final values for these cells. The Sensitivity Report also lists the final value for the parameters as well as the final value of the reduced gradient. The Limits Report is essentially used when setting the function to a specific value. [15]

For the problems in this study, I listed the data in separate columns and listed the parameters to be estimated with their starting values. I then placed the function values for each data pair minus the observed value in a separate column. In another column I placed the square of the difference found. The target cell then was the sum of the squares of the differences. It was this cell that Solver was to minimize by changing the values of the parameters. A Quasi-Newton search direction was used with automatic scaling and a tolerance of 1.0e-15. [10]

IV. F. MINPACK

Minpack is a library of FORTRAN codes which consists of software for solving nonlinear equations and nonlinear least squares problems. Minpack is freely distributed via the Netlib web site and other sources. Five algorithmic paths each include a core subroutine and driver. The algorithms proceed either from an analytic specification of the Jacobian matrix or directly from the problem functions. The paths include facilities for systems of equations with a banded Jacobian matrix, for least squares problems with a large amount of data, and for checking the consistency of the Jacobian matrix with the functions.

Minpack does return an integer variable which supplies the user with information regarding the termination of the program. If this variable, which is called ‘info’, is set equal to 0, then the user has supplied improper input parameters. For info equal to 1, the algorithm has determined that the relative error in the sum of squares is at most the quantity specified in the tolerance. If the algorithm estimates that the relative error between the current estimated values of the parameters and the actual values of the parameters is at most equal to the tolerance, then info is set equal to 2. For info equal to

3, both of the two previous conditions hold. For info equal to 4 the vector of function values is orthogonal to the Jacobian matrix to machine precision. In this case, when the linear equations in (29) are solved $p = 0$, and the algorithm cannot proceed any further. If info is set equal to 5, then the functional values have been calculated $100*(n + 1)$ times where n is the number of parameters. When info is equal to 6, no further reduction in the sum of squares is possible. If no further improvement in the estimated parameters is possible, then info is set equal to 7.

For the problems involved in this study a program and a subroutine had to be written. In the main program the parameters are declared and initialized. This program then calls the lmder1 routine. The lmder1 routine calls the subroutine written for the problem. This routine contains the data for the problem. The function values are also computed as well as the values of the partial derivatives for each data entry. The partial derivatives are calculated analytically using the supplied equations from the user. The output is printed according to the format statements given in the main program.

V. Results

The problems given in the NIST StRD dataset are provided with two separate initial starting positions for the estimated parameters. The first position, Start 1, is considered to be the more difficult as the initial values for the parameters are farther from the certified values than are the initial values given by Start 2. For this reason, one might expect that the solutions generated from Start 2 to be more accurate, or perhaps for the algorithm to take fewer iterations. It is quite interesting to note that in several cases the results from Start 2 are not more accurate based upon the minimum LRE recorded.

The critical parameter used in the comparison of these software packages is the LRE as calculated in (1). The number of estimated parameters for these problems range from two to nine. We felt it would be beneficial for the results table to be as concise as possible, yet remain useful. As a result, after running a particular package from both starting values, the LRE for each estimated parameter was calculated using Matlab 6.5. The minimum LRE for the estimated parameters from each starting position was then entered into the results table. For example, Microsoft Excel for the Gauss1 problem has an LRE given on the table of 4.7 from start position 1, and an LRE of 4.6 from start position 2. Since the Gauss1 problem has eight parameters to be estimated, the other seven parameters from each start position had a higher calculated LRE, and hence were closer to the certified values.

An entry of 0.0 in the results table is given if a software package generated estimates for the parameters but the minimum LRE was less than 1.0. For example if the minimum LRE was calculated to be 8.0e-1, rather than entering this, a 0.0 was entered. This practice was followed in an effort to be consistent with established practices. [11]

If a software package did not generate a numerical estimate for the parameters, then an entry of "NS" is entered into the results table. For example, Gnuplot on the MGH17 problem from Start 1 returned the message "BREAK Undefined value during function evaluation". Minpack returned estimates for the parameters in the Hahn1 problem as "NAN". GaussFit also returned parameter estimates for several problems as "NAN". In these instances "NS" is entered into the results table to show the reader that no numerical estimates were generated. Microsoft Excel, Nielsen's Matlab code, and SAS all generated numerical estimates for all of the problems at both starting positions.

Minimum Log Relative Error of Estimated Parameters

Problem	Start	Excel	Gnuplot	Gaussfit	HBN	Minpack	SAS
Misra1a	1	4.8	5.8	10.0	11.0	7.7	6.6
	2	6.1	5.8	10.0	10.3	7.7	7.6
Chwirut2	1	4.2	4.9	7.4	10.6	2.4	5.3
	2	4.6	4.9	8.6	9.1	2.4	5.9
Chwirut1	1	4.0	4.2	8.0	10.3	7.5	5.9
	2	4.9	4.3	8.5	10.1	7.5	4.0
Lanczos3	1	0.0	3.9	0.0	4.9	3.3	6.4
	2	0.0	3.9	7.9	5.1	3.3	5.6
Gauss1	1	4.7	5.1	8.7	6.9	8.0	0.0
	2	4.6	5.1	8.6	6.9	3.3	11.0

Problem	Start	Excel	Gnuplot	Gaussfit	HBN	Minpack	SAS
Gauss2	1	4.5	4.9	0.0	6.8	7.8	11.0
	2	4.4	4.9	0.0	6.8	7.2	9.9
DanWood	1	4.6	5.1	NS	10.2	6.6	11.0
	2	4.7	5.1	NS	8.7	6.6	11.0
Misra1b	1	4.4	5.8	0.0	10.9	2.7	7.4
	2	6.4	5.8	9.7	11.0	2.5	7.1
Kirby2	1	1.0	4.8	7.4	10.3	6.2	10.1
	2	1.9	4.9	7.9	10.4	6.2	6.8
Hahn1	1	0.0	4.0	0.0	9.5	NS	10.6
	2	0.0	4.0	0.0	9.7	NS	8.8

Problem	Start	Excel	Gnuplot	Gaussfit	HBN	Minpack	SAS
Nelson	1	0.0	0.0	0.0	0.0	0.0	0.0
	2	0.0	0.0	1.4	0.0	0.0	0.0
MGH17	1	0.0	NS	NS	0.0	7.6	0.0
	2	1.4	3.7	NS	0.0	7.5	7.3
Lanczos1	1	0.0	10.0	0.0	4.9	4.3	2.2
	2	0.0	10.0	10.0	5.8	4.3	6.3
Lanczos2	1	0.0	5.4	0.0	5.7	3.5	9.7
	2	0.0	5.4	9.1	5.3	3.5	9.5
Gauss3	1	4.3	4.8	9.2	6.5	2.4	1.4
	2	4.1	5.0	9.1	6.5	2.4	1.4

Problem	Start	Excel	Gnuplot	Gaussfit	HBN	Minpack	SAS
Misra1c	1	0.0	5.9	0.0	10.8	7.6	11.0
	2	0.0	5.9	10.0	10.2	7.6	11.0
Misra1d	1	5.2	5.8	0.0	11.0	7.6	11.0
	2	4.4	5.9	8.9	11.0	7.6	11.0
Roszman1	1	3.5	4.1	8.7	4.0	0.0	0.0
	2	0.0	5.1	8.6	4.0	0.0	0.0
ENSO	1	0.0	1.6	3.7	6.5	0.0	5.3
	2	0.0	2.2	3.7	6.6	0.0	3.0
MGH09	1	0.0	3.6	0.0	5.0	6.3	0.0
	2	5.0	3.6	0.0	5.2	6.4	7.6

Problem	Start	Excel	Gnuplot	Gaussfit	HBN	Minpack	SAS
Thurber	1	1.7	3.2	0.0	7.8	0.0	9.0
	2	1.5	4.4	6.4	7.5	0.0	8.1
BoxBOD	1	0.0	4.5	NS	9.7	0.0	10.1
	2	5.6	3.8	NS	8.6	9.1	10.4
Rat42	1	5.3	4.2	8.0	10.3	7.1	9.9
	2	5.2	4.1	8.3	11.2	7.1	8.3
MGH10	1	0.0	NS	0.0	0.0	10.8	0.0
	2	0.0	4.4	0.0	0.0	11.0	9.5
Eckerle4	1	0.0	0.0	0.0	8.1	0.0	0.0
	2	5.1	4.8	8.3	7.2	1.2	9.6

Problem	Start	Excel	Gnuplot	Gaussfit	HBN	Minpack	SAS
Rat43	1	0.0	NS	NS	0.0	6.9	10.1
	2	3.2	2.6	NS	1.3	7.0	9.6
Bennett5	1	0.0	6.4	NS	3.7	0.0	2.2
	2	0.0	6.7	NS	3.7	1.5	2.8

Note:

- NS – Software package was unable to generate any numerical solution.
- A score of 0.0 implies that the package returned a solution in which at least one parameter was accurate to less than one digit.

VI. Conclusions

The author is well aware that the conclusions reached in a study such as this are somewhat subjective in many instances. Therefore, it is with some trepidation that I state the conclusions. Certainly in the areas of accuracy and robustness the results speak for themselves. The LRE according to (1) was calculated based upon the estimates of the parameters from the various packages. This is entirely objective – yet even in this I am somewhat concerned with the implications of these results.

I can well imagine a scenario where a particular package may not be able to solve certain types of nonlinear least squares problems yet solve other types extremely well. On the other hand, as a user, such information could be extremely valuable. One would most certainly desire to have the confidence that the software package which is chosen for nonlinear least squares problems is capable of solving a variety of problems to some acceptable level of accuracy.

It is with this consideration that I write these conclusions.

VI. A. Accuracy

As stated in the introduction, we will evaluate the accuracy of the software packages in terms of the log relative error (LRE) using equation (1). Essentially the LRE will give us the number of leading digits in the estimated parameter values which correspond to the leading digits of the certified values. Again, it ought to be noted that the values given in the results table are the minimum LRE values for those problems. This is in effort to adequately evaluate the accuracy of the software package's ability to solve the problem. In other words, if a problem has five parameters to be estimated and four of the parameters are estimated accurately to seven digits, but the fifth is only accurate to one digit, it is reasonable to say that the problem was not accurately solved. On the other hand, if all five parameters were estimated to at least five digits accurately, then one could feel confident that the package had indeed solved the problem.

Nielsen's Matlab code had an average LRE score of 6.8 for the problems. For the problems this package was able to solve the starting position did not seem to be of much importance. In fact, it is quite interesting that for several problems the LRE generated using the first set of initial values is larger than the LRE generated using the second set of initial values. This is interesting because the second set of initial values is closer to the certified values of the parameter estimates. Of the twenty-three problems that the parameters were estimated correctly to at least two digits, the average LRE was 7.96. This shows us that the accuracy of the estimated parameters was very high on those problems which this package effectively solved.

GaussFit has an average LRE score of 4.9. Also, unlike Nielsen's Matlab code, GaussFit is very dependent upon the initial values given to the parameters. On eight of the problems GaussFit was unable to estimate all of the parameters to even one digit from the first starting position. From the second starting position GaussFit was able to estimate all of the parameters to over six digits correctly. This seemingly high dependence upon the starting values is a potential problem when using GaussFit for solving these nonlinear

regression problems. There is no guarantee that one can find a starting value which is sufficiently close to the solution for GaussFit to effectively solve the problem.

Gnuplot has an average LRE score of 4.6. While this is actually lower than the average LRE score for GaussFit, gnuplot is not so heavily dependent upon the starting position in order to solve the problem. Rather, much like Nielsen's code, gnuplot seems quite capable of accurately estimating the parameter values to four digits whether the starting position is close or far from the certified values.

The average LRE score for SAS is 6.46. This is quite surprising to me as I expected that SAS would be considerably more accurate. There were two problems (Nelson, and Roszman1) which SAS failed to estimate the parameters to even one digit correctly. Also surprising was the fact that on five of the problems there were rather large differences in the accuracy depending upon whether the first or second starting values were given. Again, this could be a serious problem in establishing the fact that a starting value is close enough for the package to solve the problem.

Microsoft Excel did not solve these problems well at all. The average LRE score for Excel is 2.32. Excel did perform reasonably well on the problems with a lower level of difficulty. For the eight problems with a lower level of difficulty the average LRE was 4.18. While these are probably reasonable results for these problems, we can see that for the problems with a moderate or high level of difficulty Excel did very poorly. Such results as this would cause one to have serious questions as to Excel being able to solve any particular least squares regression problem.

The Minpack library of FORTRAN codes did not perform all that well on these particular problems. The average LRE for the twenty-six problems that Minpack did solve is 4.51. Minpack was significantly less accurate than the other packages on four of the problems, Misra1b, ENSO, Thurber, and Eckerle4. On the other hand, Minpack was considerably more accurate on the MGH10 problem. Minpack does not seem to be overly dependent upon starting position as in only two of the problems was there a significant difference in the minimum LRE for the different starting positions.

VI. B. Robustness

While certainly the accuracy to which a particular software package is able to estimate the parameters is an important characteristic of the package, the ability for the package to solve a variety of nonlinear regression problems to an acceptable level of accuracy is perhaps more important to the user. This is due to fact that the user is going to have to estimate parameters from a wide range of problems. The user would certainly desire to have a level of confidence that the particular software package in use is likely to estimate those parameters to an acceptable level of accuracy.

All of this, then leads on to ask the obvious question, What is an acceptable level of accuracy? Such a question as this might elicit a variety of responses simply depending upon the nature of the study, the data, the relative size of the parameters, and many other variables which may need to be considered. For the purposes of this study we will consider an acceptable level of accuracy to be three digits.

In the chart below we will compare the various software packages by the number (and percentage) of the problems which they were able to estimate the parameters accurately to at least three digits from either starting position.

Comparison of Robustness

Package	N	P
Nielsen's Matlab code	23	85.19
GaussFit	17	62.96
Gnuplot	24	88.89
SAS	23	85.19
Excel	15	55.56
Minpack	17	62.96

Here N is the number of problems which the package accurately estimated the parameters to at least three digits. P is the percentage of the problems which the package accurately estimated the parameters to at least three digits.

It can easily be seen here that as far as the robustness of the packages is concerned there are two distinct divisions. Nielsen's Matlab code, Gnuplot, and SAS all were able to attain the 3 digit level of accuracy for over 80% of the problems. GaussFit, Excel, and Minpack, on the other hand were able to attain that level of accuracy on less than 65% of the problems.

VI. C. Features

A discussion of the features of these software packages is a rather ominous task for one to undertake. It must be remembered that some of these packages are capable of performing many more tasks than nonlinear least squares parameter estimation. On the other hand, some of these packages are designed exclusively for this.

The Matlab code written by Han Bruun Nielsen was written solely for parameter estimation of a nonlinear least squares problem. Certainly information on the actual behavior of the program in the solving of the problem is available to the user, however other statistical analysis of the estimates of the parameters is not available. This code as written is very specific to the task of estimating the parameters.

GaussFit is also designed specifically for the solution of least squares and robust estimation problems. Unlike Nielsen's Matlab code, however, GaussFit has many more features available to the user. GaussFit provides two different methods for iteration. The user can choose between Newton's method or the method of Iteratively Reweighted Least Squares by a slight change in the environment file. Two different styles of iterations through the equations are also available. The user can choose a single iteration through the forming of the equations of condition and solving the matrix, or a double iteration through the equations of condition before solving the matrix. Again, this option is available through the inclusion of a keyword in the environment file. Certainly the automatic calculations of the derivative enables the user to solve some problems that otherwise might be prohibitive. The results given by GaussFit are quite extensive as the output includes a partial triangularization of the matrix (optional); certain results of the iteration; the changes in the parameters, iteration by iteration; and the correlation matrix of reduction including a covariance column. [6]

Gnuplot is designed as a plotting package with the fit command incorporated into it. The fit command uses the Levenberg-Marquardt method with no other options available. Gnuplot does allow the user to express the standard deviation of the data and thereby change the weight of the data for use in a weighted least squares problem where the residuals are weighted before being squared. As stated above, gnuplot reports the parameter error estimates obtained from the variance-covariance matrix. The correlation matrix is also given in the output. [1]

The features of SAS are tremendous not only in number but also in potential usefulness. The PROC NLIN is but a small part of the capability of SAS. Many of the features, that is those features that were used in this study, for the NLIN procedure were covered above. Of great benefit in the SAS environment is that once the estimates of the parameters have been given the user can through the use of other procedures manipulate those estimates in virtually any number of ways. SAS has very thorough basic statistics package as well as a more than adequate plotting procedure. [14]

Excel, like SAS, is much more than a tool to solve nonlinear least squares problems. In like manner then, an added benefit of this is the potential capability to manipulate the parameter estimates within the Excel environment. Excel does not allow for many iterative methods for the solution of the nonlinear least squares problem. The user must choose either a Quasi-Newton method or the Conjugant Gradient method. [10]

Minpack, much like Nielsen's Matlab code, is very specific to what it is designed to do. As a result of this there are not many features and options available to the user. The necessity of user defined functions and particularly user defined partial derivatives may inhibit the use of Minpack in some instances, but many of the other packages would suffer from those same considerations.

Again, we must realize that comparing the features of these packages is a difficult process to carry out. These particular packages vary in design and capability substantially. In addition they vary in cost from free to quite expensive. Having said all of this, however, it is clear that the features available in the SAS package are far beyond the features available in any of the rest of the packages. The SAS environment allows the user to perform a multitude of tasks on any particular data set, of which the estimated parameters may be one.

VI. D. Ease of Use

Of the comparison criteria we are considering in this study the ease of use for the software packages is clearly the most subjective. One particular problem is that it is impossible to separate the package from the environment in which it is used. Just as any particular user has a preference in the operating system which they prefer, so too they will have a preference in the packages which they prefer to use. Just as clearly, however, there are certain items in these packages that are either user-friendly or not.

I found Nielsen's Matlab code very easy to work with. It is true that it was necessary to write separate Matlab functions to find both the function values and the Jacobian and this process was rather lengthy. (In reality, with Nielsen's code it would have been possible to simply have one function calculate both the function values and the Jacobian.) Matlab has a very easy to use 'paste special' option that allows one to copy the data from the dataset on the NIST StRD website and turn the data into vectors without having to retype all of the data. In addition to this I was able to write another Matlab function which automatically set the necessary parameters and solved all of the problems with only one running. In addition to this, since I used Matlab to calculate the LRE, I was able to have that same function calculate the LRE immediately after solving each problem. It is true that perhaps the reason I found Nielsen's code so easy to work with is more because of Matlab than the particular code, but this was an easy package with which to work.

GaussFit requires four separate files to solve each problem. The files are not difficult to write. The rather frustrating thing about the GaussFit environment is the fact that GaussFit writes over the files during the process of running the program. For this reason it was necessary to keep the files in two separate directories in order for the editing process to be easier. The program had to be run two times due to the use of two starting points. Once a few of the problems were solved with GaussFit they seemed to be rather routine for the problems that GaussFit solved.

Just like with GaussFit, once I was able to solve a few of the problems using gnuplot they were fairly routine. The data was copied into a file from the NIST website. Then the function was defined and the parameters were initialized to their starting values. The fit command was then given and the program solved the problem. As was stated earlier, the results are put into a .log file at the end of the last iteration. After gnuplot was ran for both stating values, I renamed the .log file and continued to the next problem.

SAS requires a separate program to be written for each problem. The programs varied in length depending on the number of parameters to be estimated. The program editor that is used by the SAS environment is quite unforgiving. That is, if a line was forgotten it would not allow me to insert the line in the proper position. As a result, I had to rewrite several of the programs. Another problem I ran into was that when the program is submitted (run) the program editor clears, as the output is placed in a different window. Several times I simply began writing the next program but when I submitted it the old program ran. Again, here I was forced to rewrite the program for that problem. It is true that these are things which one must learn, but they do not make for a very user-friendly package.

Excel is extremely easy to use. The only potential problem that I ran across was entering the data. Excel does not have the same 'paste special' option to express the data

as vectors as does Matlab. As a result, I cut and pasted the data from my Matlab files into the Excel worksheet. After initializing the parameters and defining the function, the rest was simply clicking with the mouse.

Minpack is written in FORTRAN, which I have not used in approximately 13 years so this takes some getting used to again. The most difficult thing about Minpack is that the data have to be hand entered into the file. With several of the problems having more than 200 data points, this is quite a tedious, time-consuming job.

Several of these packages were what I would consider to be easy to use. Certainly Excel, gnuplot and Nielsen's Matlab codes were the easiest with GaussFit close behind. SAS is rather programming intensive and with certain keywords which must be used in a certain order. Additionally, the program editor in SAS is difficult to use until one is familiar with it. The entering of the data for the Minpack programs was the one item which made it rather a challenge to use.

VII. Bibliography

1. Crawford, Dick "Gnuplot Manual", http://www.comnets.rwth-aachen.de/doc/gnu/gnuplot37/gnuplot.html .

2. Davis, Paul "Levenberg – Marquardt Methods and Nonlinear Estimation" *SIAM News*, Vol. 26, No. 6, October 1993.

3. Dennis, J.E., Gay, D.M., and Welch, R.E. "An adaptive nonlinear least – squares algorithm", NBER working paper 196, M.I.T./Center for Computational Research in Economics and Management Science, Cambridge, Mass., August 1977.

4. Engineering Statistics Handbook, 4.1.4.2 Nonlinear Least Squares Regression, http://www.itl.nist.gov/div898/handbook/pmd/section1/pmd142.htm, 1-3.

5. Hiebert, K.L. "An Evaluation of Mathematical Software That Solves Nonlinear Least Squares Problems", *ACM Transactions on Mathematical Software*, Vol. 7, No. 1, March 1981, 1-16.

6. Jefferys, W.H., Fitzpatrick, M.J., McArthur, B.E., McCartney, J.E. "GaussFit: A System for Least Squares and Robust Estimation Users Manual", University of Texas, 1998.

7. Johnson, R.A., Wichern, D.W. "Applied Multivariate Statistical Analysis", Fifth Edition, Prentice-Hall 2002, 354-377.

8. Kitchen, A.M., Drachenberg, R., Symanzik, J. "Assessing the reliability of web-based statistical software", Department of Mathematics and Statistics, Utah State University

9. Marquardt, D.W. "An Algorithm for Least Squares Estimation of Nonlinear Parameters", *Journal for The Society of Industrial and Applied Mathematics*, 11, 431-441.

10. Mathews, M., Seymour S., "Excel for Windows: The Complete Reference", Second Edition, McGraw-Hill Inc., 1024-1108.

11. McCullough, B.D., "Assessing reliability of statistical software: Part I:, *The American Statistician*, November 1998, Vol. 52, No. 4, 358-366.

12. More, J.J., Garbow, B.S., and Hillstrom, K.E., "Testing unconstrained optimization software. Rep. TM-324, Appl. Math Div., Argonne National Lab., Argonne, IL., 1978.

13. Nash, S.G., Sofer, A., "Linear and Nonlinear Programming" McGraw – Hill New York (1996), pgs 409-423.

14. SAS Users Guide, Version 6, Fourth Edition Volume 2, SAS Institute Inc. Cory, NC, 1990, 1136-1193.

15. Seber, G. A. F., Wild, C. J., "Nonlinear Regression", John Wiley & Sons New York (1989), pgs 21-33, 701-702.