On the Numerical Accuracy of Mathematica 5.0 for Doing Linear and Nonlinear Regression

Marc Nerlove

This article presents the results of performing the linear and nonlinear regressions used as benchmarks by the National Institute of Standards and Technology (NIST) with *Mathematica* 5.0. The performance is nearly flawless.

"Here is the guess of their true strength and forces
By diligent discovery;... "
King Lear, V. i. 52

■ 1. Introduction

There is a small but growing literature on the numerical accuracy of various computer programming packages. A major contributor to the assessment of the accuracy of econometric software, as well as statistical add-ons to third-level programming languages widely used by econometricians, such as GAUSS, MATLAB, Maple, and *Mathematica*, has been Bruce D. McCullough ([1] and the references cited therein).

McCullough and Wilson [2, 27] assessed the reliability of the statistical procedures in Excel 97 for estimation of both linear and nonlinear regression, random number generation, and the calculation of cumulative distributions and concluded that "Excel's performance in all three areas is found to be inadequate. Persons desiring to conduct analyses of statistical data are advised not to use Excel." McCullough made a complete assessment of all of *Mathematica* 4.0's then extant statistical functions [3]. In 2000, I made a similar assessment for ordinary least squares (OLS) linear regression analysis using Excel 2000 and the Numerical Algorithms Group's (NAG's) Statistical Add-Ins for Excel software (1). I found that the "NAG-enhanced" version of Excel did not do significantly better than the non-enhanced version. In the same paper, I also compared the performance of *Mathematica* 4.1.

The basis for McCullough's assessments as well as mine were comparisons with the benchmark results for the Statistical Reference Datasets (StRD) that are available from the NIST (2).

My comparison of Excel 2000 with the benchmark linear regression analyses produced results that did not differ significantly from those McCullough obtained for Excel 97. Notwithstanding the excellent reputation of NAG's software, the performance of the Excel add-on is not superior to Excel. For cases in which Excel performs well, the NAG add-on also does well, but when Excel performs unsatisfactorily, the NAG add-on does poorly as well. However, often the results differ from the certified results in a different direction from those obtained directly in Excel. In the case of linear regression, on the other hand, Excel on its own does not do much worse than many other packages analyzed by McCullough and others. My comparisons at the time suggested that McCullough's judgment, at least with respect to linear regression, may have been unduly critical; however, I do agree that the statistical functions in Excel should not be used with or without NAG enhancement.

Vinod [4, 211] undertook a careful analysis of the statistical functions in GAUSS for Windows Version 3.2.27, also using comparisons with the StRD. He found "… that the algorithms used by GAUSS sometimes fail rather badly to provide accurate results. We blame both the algorithms used and the language itself." I have not yet undertaken a similar reassessment of the more recent and allegedly significantly improved versions of GAUSS.

McCullough [5, 152] applied similar methodology to SAS, SPSS, and S-Plus, which I call fourth-level programming languages or statistical packages. With respect to univariate statistics, analysis of variance, and linear regression, McCullough reported that "All [SAS] results are quite accurate," in terms of six digits of accuracy or better, although SAS failed to deliver any solution in one of the worst cases. Even though SAS failed to achieve convergence in a number of cases, the results for nonlinear regression were similar. The performance of all three packages with respect to random number generation and the calculation of probabilities for statistical distributions was notably poorer than for OLS linear regression.

McCullough [3] analyzed the accuracy of *Mathematica* 4.0's statistics add-on package (3). *Mathematica* is not restricted as to precision; however, precision may be specified. The default is to use machine arithmetic, which is typically 16 digits of precision in internal calculations (a typical value of `$MachinePrecision` is $53 \log_{10} 2$ or approximately 16). Of course this does not guarantee 16 digits of precision in the final result. Note that accuracy and precision are not the same as defined in *Mathematica*. A higher level of precision may be attained by setting `$MinPrecision` above 16 or by setting `WorkingPrecision→n`, where *n* is a number greater than `MachinePrecision`. Only a small number of *Mathematica* functions support the `WorkingPrecision` option. Moreover, when software other than *Mathematica* is used, increasing the level of precision in this way may not be possible. This is because within the code the computation has been carried out in floating-point notation. No matter how many degrees of arbitrary precision are requested, the result is still contaminated with the errors from floating-point arithmetic. Arbitrary precision only works well when the computation has been carried out uncontaminated with floating-point error. In *Mathematica* it is possible to carry out the computation with numbers expressed in rational form, which yields exact results, but this still will not avoid the problem of inverting an ill-conditioned matrix.

At default precision, *Mathematica* is quite accurate although McCullough obtained no solution in one StRD case of OLS linear regression and one in an StRD case of nonlinear regression. When `$MinPrecision` is raised to 20 digits, McCullough reports that astonishing, almost perfect, accuracy, as compared with the NIST benchmark results, is obtained in all cases considered. Actually, *Mathematica*'s default level of precision gives exact matches to the benchmark results in most cases, but the results *Mathematica* displays are rounded to six places and thus may not look quite the same. To display the results to more places, use `NumberForm`. The increase in computational time using a higher level of precision than default in direct, and therefore symbolically pure, computation of the OLS estimates appears to be negligible no matter how high the level of precision is specified. Although it is certainly true that increased accuracy may not matter for the conclusions to be drawn from a statistical analysis, it may also happen that rounding errors in the course of an otherwise straightforward calculation may result in wildly different results or that no solution is found. If the cost of obtaining accurate results even in problematic cases is not great, it is certainly prudent to obtain what increased accuracy may be possible. If a preprogrammed package is used, as it is in this article, it may not be possible to control for rounding error because of the original coding.

In this article, I discuss the details of my own comparisons of *Mathematica* 5.0's standard statistical add-on package. McCullough employs summary measures, which must of necessity leave out a lot of relevant detail in the case of many parameters, but I present a full range of results. A separate section deals with the more computationally difficult benchmark cases for nonlinear regression.

In the next section, I describe the NIST StRD benchmarks, particularly the 11 designed for benchmarking OLS linear regression (i.e., linear in the parameters to be estimated, so polynomial regression is not excluded) and the 27 nonlinear benchmark datasets. I also discuss the way in which the NIST certified results have been obtained for the linear regression StRD set. In Section 3, I discuss the general problems of nonlinear regression analysis and the difficulty obtaining appropriate significance tests short of computing the parameter estimates by maximum likelihood (ML). In Section 4, I summarize the results of using Version 5.0 of *Mathematica*'s `Statistics'LinearRegression'`standard add-on package and compare these with the StRD benchmark results. Essentially the same results may be obtained directly by writing out the standard textbook formulae in *Mathematica*, although these formulae are not the best way of carrying out the requisite numerical calculations. Details of the numerical results are presented in Appendices A and B, in which tabular comparisons are made of the NIST StRD benchmark results with those obtained using *Mathematica*'s `Statistics`LinearRegression`` standard add-on package and those using the `NonlinearRegress` option in the `Statistics`NonlinearFit`` standard add-on package. The appendices are part of the electronic version of this article, but are not included in the printed version.

■ 2. The NIST StRD Datasets and Certified Results for Linear and Nonlinear Regression

The NIST describes the statistical benchmark program on their website as follows.

"Currently datasets and certified values are provided for assessing the accuracy of software for univariate statistics, linear regression, nonlinear regression, and analysis of variance. The collection includes both generated and 'real-world' data of varying levels of difficulty. Generated datasets are designed to challenge specific computations. These include the classic Wampler datasets for testing linear regression algorithms and the Simon & Lesage datasets for testing analysis of variance algorithms. Real-world data include challenging datasets such as the Longley data for linear regression, and more benign datasets such as the Daniel & Wood data for nonlinear regression. Certified values are 'best-available' solutions. The certification procedure is described in the web pages for each statistical method.

"Datasets are ordered by level of difficulty (lower, average, and higher). Strictly speaking the level of difficulty of a dataset depends on the algorithm. These levels are merely provided as rough guidance for the user. Producing correct results on all datasets of higher difficulty does not imply that your software will pass all datasets of average or even lower difficulty. Similarly, producing correct results for all datasets in this collection does not imply that your software will do the same for your particular dataset. It will, however, provide some degree of assurance, in the sense that your package provides correct results for datasets known to yield incorrect results for some software" (2).

□ (a) The NIST StRD Datasets for Linear Regression

The particular sets for linear regression analysis are summarized in Table 1. The problem of linear regression and the methods employed by the NIST, as well as *Mathematica*, are discussed in Section 3 (a).

Dataset Name	Level of Difficulty	Model Class	Number of Parameters	Number of Observations	Source
Norris [6]	Lower	Linear	2	36	Observed
Pontius [7]	Lower	Quadratic	3	40	Observed
NoInt1 [8]	Average	Linear	1	11	Generated
NoInt2 [8]	Average	Linear	1	3	Generated
Filip [9]	Higher	Polynomial	11	82	Observed
Longley [10]	Higher	Multilinear	7	16	Observed
Wampler1 [11]	Higher	Polynomial	6	21	Generated
Wampler2 [11]	Higher	Polynomial	6	21	Generated
Wampler3 [11]	Higher	Polynomial	6	21	Generated
Wampler4 [11]	Higher	Polynomial	6	21	Generated
Wampler5 [11]	Higher	Polynomial	6	21	Generated

Source: www.itl.nist.gov/div898/strd/ils/ils.shtml

Table 1. StRD benchmark datasets for linear regression.

The following edited comments from the NIST website describe these datasets in more detail.

> "Both generated and 'real-world' data are included. Generated datasets challenge specific computations and include the Wampler data developed at NIST (formerly NBS) in the early 1970s. Real-world data include the challenging Longley data, as well as more benign datasets from our statistical consulting work at NIST.
>
> "… Two datasets are included for fitting a line through the origin. We have encountered codes that produce negative R-squared and incorrect F-statistics for these datasets. Therefore, we assign them an 'average' level of difficulty. Finally, several datasets of higher level of difficulty are provided. These datasets are multicollinear. They include the Longley data and several NIST datasets developed by Wampler.
>
> "… Certified values are provided for the parameter estimates, their standard deviations, the residual standard deviation, R-squared, and the standard ANOVA table for linear regression. Certified values are quoted to 16 significant digits and are accurate up to the last digit, due to possible truncation errors.
>
> "… If your code fails to produce correct results for a dataset of higher level of difficulty, one possible remedy is to center the data and rerun the code. Centering the data, that is, subtracting the mean for each predictor variable, reduces the degree of multicollinearity. The code may produce correct results for the centered data. You can judge this by comparing predicted values from the fit of centered data with those from the certified fit."
> (www.itl.nist.gov/div898/strd/ils/ils_info.shtml).

□ (b) The NIST StRD Datasets for Nonlinear Regression

The particular sets for nonlinear regression analysis are summarized in Table 2. The problem of nonlinear regression and the methods employed by the NIST, as well as *Mathematica*, are discussed in Section 3 (b).

Dataset Name	Level of Difficulty	Model Classification	Number of Parameters	Number of Observations	Source
Misra1a [12]	Lower	Exponential	2	14	Observed
Chwirut2 [13]	Lower	Exponential	3	54	Observed
Chwirut1 [13]	Lower	Exponential	3	214	Observed
Lanczos3 [14]	Lower	Exponential	6	24	Generated
Gauss1 [15]	Lower	Exponential	8	250	Generated
Gauss2 [15]	Lower	Exponential	8	250	Generated
DanWood [16]	Lower	Miscellaneous	2	6	Observed
Misra1b [12]	Lower	Miscellaneous	2	14	Observed
Kirby2 [17]	Average	Rational	5	151	Observed
Hahn1 [18]	Average	Rational	7	236	Observed
Nelson [19]	Average	Exponential	3	128	Observed
MGH17 [20]	Average	Exponential	5	33	Generated
Lanczos1 [14]	Average	Exponential	6	24	Generated
Lanczos2 [14]	Average	Exponential	6	24	Generated
Gauss3 [15]	Average	Exponential	8	250	Generated
Misra1c [12]	Average	Miscellaneous	2	14	Observed
Misra1d [12]	Average	Miscellaneous	2	14	Observed
Roszman1 [21]	Average	Miscellaneous	4	25	Observed
ENSO [22]	Average	Miscellaneous	9	168	Observed
MGH09 [23]	Higher	Rational	4	11	Generated
Thurber [24]	Higher	Rational	7	37	Observed
BoxBOD [25]	Higher	Exponential	2	6	Observed
Rat42 [26]	Higher	Exponential	3	9	Observed
MGH10 [27]	Higher	Exponential	3	16	Generated
Eckerle4 [28]	Higher	Exponential	3	35	Observed
Rat43 [26]	Higher	Exponential	4	15	Observed
Bennett5 [29]	Higher	Miscellaneous	3	154	Observed

Source: www.itl.nist.gov/div898/strd/nls/nls_main.shtml

Table 2. StRD benchmark datasets for nonlinear regression.

The following edited comments from the NIST website describe these datasets in more detail.

"… Hiebert [30] notes that 'testing to find a "best" code is an all but impossible task and very dependent on the definition of "best."' Whatever other criteria are used, the test procedure should certainly attempt to measure the ability of the code to find solutions. But nonlinear least squares regression problems are intrinsically hard, and *it is generally possible to find a dataset that will defeat even the most robust codes. So most evaluations of nonlinear least squares software should also include a measure of the reliability of the code, that is, whether the code correctly recognizes when it has (or has not) found a solution*. The datasets provided here are particularly well suited for such testing of robustness and reliability. [Emphasis added.]

"… both generated and 'real-world' nonlinear least squares problems of varying levels of difficulty [are included]. The generated datasets are designed to challenge specific computations. Real-world data include challenging datasets such as the Thurber problem, and more benign datasets such as Misra1a. The certified values are 'best-available' solutions, obtained using 128-bit precision and confirmed by at least two different algorithms and software packages using analytic derivatives.

"… For some of these test problems, however, it is unreasonable to expect the correct solution from a nonlinear least squares procedure when finite difference derivatives are used…. These difficult problems are impossible to solve correctly when the matrix of predictor variables is only *approximate* because the user did not supply analytic derivatives.

"… the datasets have been ordered by level of difficulty (lower, average, and higher). This ordering is meant to provide rough guidance for the user. Producing correct results on all datasets of higher difficulty does not imply that your software will correctly solve all datasets of average or even lower difficulty. Similarly, producing correct results for all datasets in this collection does not imply that your software will do the same for your own particular dataset. It will, however, provide some degree of assurance, in the sense that your package provides correct results for datasets known to yield incorrect results for some software.

"The robustness and reliability of nonlinear least squares software depends on the algorithm used and how it is implemented, as well as on the characteristics of the actual problem being solved. Nonlinear least squares solvers are particularly sensitive to the starting values provided for a given problem. For this reason, we provide three sets of starting values for each problem: the first is relatively far from the final solution; the second relatively close; and the third is the actual certified solution.

"… sometimes good starting values are not available. For testing purposes, therefore, it is of interest to see how a code will perform

> when the starting values are not close to the solution, even though such starting values might be ridiculously bad from a practical standpoint. In general, it can be expected that a particular code will fail more often from the starting values far from the solution than from the starting values that are relatively close. How serious it is that a code fails when using starting values far from the solution will depend on the types of problems for which the code will be employed." (www.itl.nist.gov/div898/strd/nls/nls_info,shtml).

In my assessment of *Mathematica* 5.0, I use only the NIST starting values, which are different from the certified results, except in one instance. This is the MGH10 dataset, for which *Mathematica* gives no results for both the first and second starting values. When the certified results are used as the starting value, *Mathematica* crashes. The problem of choosing good starting values is central to nonlinear regression. As discussed later, I believe that such a choice should not be made automatically, but rather involve some prior investigation.

■ 3. Methods for Linear and Nonlinear Regression

□ (a) Linear Regression

The general statistical model assumed for the linear least-squares regression problems is

$$y = X\beta + \epsilon,$$

where y denotes the response (dependent) variable, β denotes the vector of p unknown parameters to be estimated, n is the number of observations, and X denotes the n by p matrix of predictor (independent) variables. The specific functional form for each dataset is different from set to set and is specified in the certified results for that set, summarized in Appendix A. Unless the constant term in the regression is suppressed, X is assumed to contain a column of ones, the coefficient of which is the nonzero intercept. If we want to make inferences about β or how well the relationship fits the data, we have to make some assumptions about the joint distribution of X and ϵ. The usual minimal assumptions are that ϵ has a distribution with variance-covariance matrix $\sigma^2 I$ independently of X. In this case, the least-squares estimates of β,

$$b = (X'X)^{-1} X' y,$$

have certain optimal properties, and the estimated standard errors of b are

$$se = \sqrt{s^2 (X'X)^{-1}}\,,$$

where

$$s^2 = \frac{e'e}{n-k}$$

is an estimate of σ^2 and

$$e = y - X\,b$$

are the calculated residuals from the regression. The measure R^2 of goodness-of-fit is defined as

$$R^2 = 1 - \frac{e'e}{y'y - n\,\overline{y}}$$

when an intercept (frequently represented by including a column of ones in the matrix X) is included and defined as

$$R^2 = 1 - \frac{e'e}{y'y}$$

when no intercept is included (e.g., NoInt1 and NoInt2 [8]). Here is what the NIST says about the method employed for linear regression.

> "For all datasets, multiple precision calculations (accurate to 500 digits) were made using the preprocessor and FORTRAN subroutine package of Bailey (1995). Data were read in exactly as multiple precision numbers and all calculations were made with this very high precision. The results were output in multiple precision, and only then rounded to fifteen significant digits. These multiple precision results are an idealization. They represent what would be achieved if calculations were made without roundoff or other errors. Any typical numerical algorithm (i.e., not implemented in multiple precision) will introduce computational inaccuracies, and will produce results which differ slightly from these certified values." (See the certification methodology description for each dataset at www.itl.nist.gov/div898/strd/lls/lls.shtml, accessed 07/12/03.)

Although not generally advisable, it is perfectly possible to simply write out these equations in *Mathematica* and solve a particular problem, since standard methods for solving for inverses of well-conditioned matrices may not give accurate answers for ill-conditioned matrices. This was what, in fact, went wrong with most of the extant computer packages for doing ordinary least-squares regression when Longley [10] did his famous study. *Mathematica*, however, is very sophisticated about the way it inverts matrices and, indeed, gets this problem right, albeit with a warning message in the program exhibited for the computation of `invX1`.

Daniel Lichtblau comments on the difference between these two methods of doing OLS linear regression as follows: "What we [*Mathematica*] actually work with [in the add-on package, `Statistics`LinearRegression``] are singular values. In effect, this is a method to compute a pseudo-inverse (which we do not actually do, unless necessitated by the various report option settings). In contrast, matrix inversion will be done using Gaussian elimination, with appropriate warnings if the matrix is found to be ill conditioned."

Here is my "direct" program.

```
longley = Import["Longley.txt", "Table"];
data = Rest[longley];
{n, k} = Dimensions[data];
y = Map[First, data];
X = Map[Rest, data];
X1 = Map[Prepend[#, 1] &, X];
invX1 = Inverse[Transpose[X1].X1];
b = invX1.(y.X1);
res = y - b.Transpose[X1];
s2 = res.res / (n - k);
se = Sqrt[s2 * Tr[invX1, List]];
ȳ = Apply[Plus, y] / n;
R2 = 1 - (res.res) / ((y - ȳ).(y - ȳ));
```

The reader, of course, will have to fill in the exact location of the Longley.txt file, which is copied as a .txt file in Appendix C.

The *Mathematica* parameter estimates and standard errors compared with the benchmark results are:

Parameters	*Mathematica* Estimate	Benchmark Estimate	*Mathematica* SE	Benchmark SE
β_0	-3.48226×10^6	-3.48226×10^6	890420.	890420.
β_1	15.0619	15.0619	84.9149	84.9149
β_2	−0.0358192	−0.0358192	0.033491	0.033491
β_3	−2.02023	−2.02023	0.4884	0.4884
β_4	−1.03323	−1.03323	0.214274	0.214274
β_5	−0.0511041	−0.0511041	0.226073	0.226073
β_6	1829.15	1829.15	455.478	455.478

And of R^2 and s^2:

Parameters	*Mathematica* Values	Benchmark Values
s^2	92936.	92936.
R^2	0.995479	0.995479

If I were to exhibit the results to 15 digits, the *Mathematica* and benchmark results would also coincide exactly. Section 3.1.4 of *The Mathematica Book* [31] gives a good discussion of the way in which *Mathematica* handles the problem of numerical precision in general and floating-point arithmetic in particular. So when doing linear least-squares regression it does not appear to matter whether I use my own code or *Mathematica*'s `Regress` function.

I have not been able to find documentation as to exactly how *Mathematica* performs a linear least-squares regression with its `Regress` function, other than the full code for `LinearRegression` in `AddOns/StandardPackages/Statistics'LinearRegression'`. (See Daniel Lichtblau's earlier comments.) Needless to say, it is not so easy for a nonprogrammer to read this code. It appears that *Mathematica* uses the same computationally efficient and accurate methods to solve the usual least-squares normal equations as it does to solve any system of linear equations or to invert a numerically specified square matrix. See Section A.9.4 of *The Mathematica Book* [31].

Comparisons of the results from `Regress` with the NIST benchmarks for each of the 11 NIST SdRD linear regression examples are presented in summary form in Appendix A. The results presented for *Mathematica* are simply those *Mathematica* prints out. The reader, however, should be aware that these are not the actual results of *Mathematica*'s computation, which are all carried out to 16 digits of precision, but rather a rounded form for ease of visual presentation. The full 16 digits can easily be obtained by using `NumberForm[expr, {n, f}]`, which prints approximate real numbers having n digits, with f digits to the right of the decimal point.

□ (b) Nonlinear Regression

Following the NIST description, the statistical model assumed for nonlinear regression in the benchmark calculations is the so-called generic univariate case:

$$y = f\,(x;\ \beta) + \epsilon,$$

where y denotes the response (dependent) variable, x denotes the predictor (independent) variables, and β (unsubscripted) the p unknown parameters to be estimated. The specific functional forms assumed differ from one benchmark case to the next. They are reported in Appendix B, in which comparisons between the results obtained by *Mathematica* and those obtained by the NIST are presented. The NIST methodology is described as follows.

> "The certified values for the nonlinear least-squares regression problems were obtained using 128-bit precision with the reported results confirmed by at least two different algorithms and software packages using analytic derivatives.
>
> "The certified values for the estimates, $b = (b_1, b_2, \ldots, b_p)$ of the true model parameters β are those produced by the smallest sum of squares, i.e.,
>
> $$b = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} [y_i - f(x_i;\ \beta)]^2 \right\},$$
>
> where n denotes the number of observations. Under the assumption that
>
> $$\epsilon_i \equiv [y_i - f(x_i;\ \beta)] \sim N(0,\ \sigma^2),$$
>
> it follows that these are the maximum-likelihood estimators.

"The certified values for the standard deviations of the estimates of the model parameters are the square roots of the diagonal elements of the asymptotic covariance matrix,

$$V = s^2[\mathcal{J}'\,\mathcal{J}]^{-1},$$

where

$$s = \sqrt{\frac{\min_\beta\left\{\sum_{i=1}^n [y_i - f(x_i;\,\beta)]^2\right\}}{n-p}} = \sqrt{\frac{\min_\beta\left\{\sum_{i=1}^n [y_i - f(x_i;\,b)]^2\right\}}{n-p}},$$

$\mathcal{J}$ denotes the Jacobian matrix with *ij*th element

$$\mathcal{J}_{ij} = \frac{\partial f(x_i;\,\beta)}{\partial \beta_i},\ i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, p,$$

evaluated at the current values of the parameters $b_1, b_2, \ldots, b_p$, and n and p denote the number of observations and the number of parameters, respectively.

"The certified value of the residual sum of squares is defined by

$$\mathrm{SS} == \min_\beta\left\{\sum_{i=1}^n [y_i - f(x_i;\,\beta)]^2\right\} = \min_\beta\left\{\sum_{i=1}^n [y_i - f(x_i;\,b)]^2\right\}$$

where n denotes the number of observations."

The benchmark estimate of σ^2 is thus $\mathrm{SS}/(n-p)$ (www.itl.nist.gov/div898/strd/nls/nls_info.shtml).

Under the assumption $\epsilon_i \equiv [y_i - f(x_i;\,\beta)] \sim N(0,\,\sigma^2)$, it is thus possible to derive all of these estimates by maximum likelihood. In this case, the standard errors of the estimates are obtained asymptotically as the square roots of the diagonal elements of the inverse of the information matrix (the Hessian of the likelihood function evaluated at the maximizing point). (See the standard econometric presentations of nonlinear regression methods in Davidson and MacKinnon [32] and Greene [33].) Indeed, it is in some ways a more general method of analysis, since ML may be applied in the case of more general models.

Before proceeding, it is worth noting a few limitations of the standard univariate nonlinear regression model, $y = f(x;\,\beta) + \epsilon$. According to Davidson and MacKinnon [32, 42], "The feature that distinguishes regression models from all other statistical models is that the only way in which randomness affects the dependent variable is through an *additive* error term or disturbance." However, this may be less of a restriction than you might at first think. For example, suppose a multiplicative disturbance $y = f(x;\,\beta) * \epsilon$; then a simple logarithmic transformation suffices to return things to additivity: $\mathrm{Log}[y] = \mathrm{Log}[f(x;\,\beta)] + \mathrm{Log}[\epsilon]$. Of course any distributional assumptions we may have made concerning ϵ may have to be modified. Independence of the regressors, x, and the disturbance, or at least lack of correlation, is an assumption common to both linear and nonlinear regression. A problem arises, however, if the appropriate transformation depends on parameters to be estimated. For example, a common generalization is a transformation of the dependent variable involving another parameter to be estimated:

$$g(y;\ \theta) = f(x;\ \beta) + \epsilon.$$

In this case, the distribution of the ys, given the xs, will no longer be the same as the distribution of the ϵs, and least squares may be problematic and generally will no longer yield the same estimates as maximum likelihood. Maximum likelihood is still an effective method, however, since the distribution of the ys, given the xs, may still be found from the distribution of the ϵs by multiplying by the Jacobian of the transformation, $g(y;\ \theta)$. A more serious limitation is the necessity of specifying the functional form of f. Whereas, in the case of linear regression we hardly ever think about alternative functional forms, except perhaps to worry about interactions, the very fact that we are considering nonlinear regression underscores a concern about the appropriate functional form. In recent years, there has been an efflorescence of work on nonparametric methods explicitly designed to cope with this problem. (See, especially Haerdle [34], and Pagan and Ullah [35]).

There is a extensive literature on nonlinear regression, including standard texts, such as Gallant [36], Bates and Watts [37], and Seber and Wild [38], each with somewhat different emphasis. Most emphasis falls on computational issues and problems of parametric inference. A very good brief expository introduction is Gallant [39].

Most algorithms for computing nonlinear regression parameters are based on Hartley's [40] modified Gauss–Newton method (Hartley and Booker, [41]) or as extended and modified by Marquardt [42] following a suggestion by Levenberg [43]. There is an important difference between maximizing/minimizing functions in general and the problem of finding the minimum of an analytically specified sum of squares (see the previous SS). In general, we can evaluate only the function itself analytically and perhaps its gradient. Using only these two pieces is the basis for the method of steepest descent, sometimes also known as the Gauss–Newton method. But, in the case of nonlinear least squares, we know a great deal more: we know the Hessian matrix. Hartley's insight was to exploit this information explicitly, rather than build it up piecemeal during the iterative process of descent. His modified Gauss–Newton method is called the inverse Hessian method. Levenberg's insight was to recognize that simple steepest descent is much faster and more accurate when you are close to the minimum and to suggest separating the problem of getting into the neighborhood of the minimum from "zipping in." Marquardt's contribution was an elegant algorithm designed to vary smoothly between the two methods as the minimum point was approached. *Mathematica*'s `NonlinearRegress` uses the Levenberg–Marquardt algorithm by default, but also allows you to choose among several other algorithms for function minimization.

In [30, 44], Hiebert did a careful evaluation of existing programs for doing nonlinear least squares and for solving systems of nonlinear equations (a harder problem). One surprising conclusion he drew [30, 15] was that, "On the basis of this testing and these specific implementations of the Levenberg–Marquardt and the augmented Gauss–Newton methods, one method does not appear to be superior to the other." Perhaps, for this reason, the NIST uses a variety of methods, each considered appropriate for the problem at hand. However, the NIST is not more explicit about the methods used in specific cases than the statements quoted earlier.

Standard-error calculations and inference for nonlinear least-squares estimates are all based on asymptotic results (Hartley and Booker, [41]; Jennrich, [45]) and deftly summarized by Gallant [39]. These are the basis for the NIST calculations detailed earlier. *Mathematica* does not say how its calculations are carried out in the description of the `NonlinearRegress` function; however, the code is available in the `Statistics`NonlinearRegress`` package, and it appears that the same formulae are used.

Comparisons of the results from `NonlinearRegress` with the NIST benchmarks for each of the 27 NIST StRD nonlinear regression examples with their published starting values are presented in summary form in Appendix B. The NIST actually ran all computations using not only these two starting values but also the final benchmark answers as starting values because some nonlinear regression algorithms have a problem if you start from the right answer. However, in a few cases, I also tried starting at the benchmark answers: in the case of MGH10, as noted earlier, and for those few starting values for which *Mathematica* did not converge: Lanczos2; MGH17; MGH09; as well as MGH10. However, as noted by the NIST, MGH10 is a particularly difficult problem: "This problem was found to be difficult for some very good algorithms. There is a local minimum at (+inf, -14.07 ... , -inf, -inf) with final sum of squares 0.00102734." [46, 17] Lanczos3 is also a rather peculiar function, but *Mathematica* generally performs quite well for chosen starting values.

The crucial problem of choosing good starting values is discussed extensively by Bates and Watts [37, 72–76], and, more briefly, by Seber and Wild [38, 665–666]. Bates and Watts suggest using a linearized form of the expectation function, while Seber and Wild suggest grid search or randomized starting values. Ratkowsky [26] gives a much more extensive discussion. When only one or two parameters are involved, a graphical approach works wonders. However, when three or more parameters are involved, special techniques are required and the difficulties are formidable. I have been unable to ascertain how the NIST chose their starting values, which are always one quite close to and another far away from the final answer, but *Mathematica* does offer an alternative. Before invoking `NonlinearRegress`, you could use `NMinimize` with no starting values (or some known to cause problems) assumed to find the minimum of the appropriate sum of squares. This is a very elaborate and well-documented method in *Mathematica* for finding global minima. Such more sophisticated methods may not be very efficient compared to the algorithm *Mathematica* employs in `Nonlin`⋱ `earRegress`, but may work in such difficult cases. In computing the results reported here, I avoid the problem of choosing starting values by simply using those reported by the NIST.

4. Discussion of the Results Using *Mathematica* 5.0

(a) Linear Regression

There is really not much to say about the performance of *Mathematica* in doing linear regression. *Mathematica* is remarkably accurate with only one negligible but puzzling anomaly: In the cases of Wampler1 and Wampler2, both fifth degree polynomials with artificially constructed data, the true SEs are 0, as is the correct estimate of s^2. However, *Mathematica* delivers very small numbers of the order of 10^{-10} to 10^{-28}. While this is not a practical problem of any significance, some explanation is needed in view of *Mathematica*'s otherwise stellar performance. I would attribute it to *Mathematica*'s use of variable-precision floating-point arithmetic [47], but the matter clearly requires further looking into.

(b) Nonlinear Regression

Nonlinear regressions present a greater challenge. In all but five out of 27 difficult nonlinear regression problems, *Mathematica* converged to the benchmark values within the default number of iterations, `MaxIter=100` for both of the two start values given by the NIST. But there were serious problems with five of the problems, including one case in which no convergence was obtained even starting from the final benchmark result. However, in all five cases, informative error messages were given by *Mathematica* and no spurious results were presented. Here is a summary.

MGH17

$$y = \beta_1 + e^{-x\beta_4}\,\beta_2 + e^{-x\beta_5}\,\beta_3 + \epsilon$$

As noted earlier, this is an exceptionally difficult problem given the observations on y and x. Start 1 produced an overflow in the computation. There were no problems, however, encountered in obtaining the benchmark results from start 2.

Lanczos2

$$y = e^{-x\beta_2}\,\beta_1 + e^{-x\beta_4}\,\beta_3 + e^{-x\beta_6}\,\beta_5 + \epsilon$$

Note that this is the same functional form as MGH17 without a constant term. In both examples the data has been generated by the NIST. No convergence was obtained for the default number of iterations from start 1 with an error message to that effect. Convergence was, however, obtained for `MaxIter=200` from start 1 and from start 2 for `MaxIter=100` to the correct benchmark results.

MGH09

$$y = \frac{\beta_1\,(x^2 + x\,\beta_2)}{x^2 + x\,\beta_3 + \beta_4} + \epsilon$$

No convergence was obtained from start 1 even by increasing the number of iterations to 500. The following error message suggests that the function is extremely flat, at least in one direction.

```
FindFit::"lstol":"The line search decreased the step size to within tolerance
specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient
decrease in the norm of the residual. You may need more than MachinePrecision
digits of working precision to meet these tolerances."
```

Convergence to the benchmark results for `MaxIter=100` from start 2 was obtained without any problems.

MGH10

$$y = \frac{\beta_1}{1 + e^{\frac{\beta_2}{x+\beta_3}}} + \epsilon$$

No convergence was obtained from start 1 or from start 2 for 10,000 iterations. Starting from the benchmark results produced an abnormal program termination with a message that the fitting algorithm failed.

Bennett5

$$y = \beta_1 \, (x + \beta_2)^{-\frac{1}{\beta_3}} + \epsilon$$

For start 1, convergence was obtained only for 10,000 iterations. Although the benchmark coefficients and standard errors were obtained, the benchmark estimate of residual variance was not. For start 2, convergence to the benchmark results within 100 iterations was obtained.

Note that *Mathematica*'s `NonlinearRegress` is extremely fast. The time to run a regression once the data is set up and the command given is imperceptible for the default number of 100 iterations. Even for 10,000 iterations, the running time is barely perceptible on a Dell Workstation, PWS 340, with an Intel® Pentium chip.

Mathematica's `NonlinearRegress` is an extremely fast and accurate algorithm when it works. When it does not, it delivers error messages rather than spurious results. Presently, there is no provision in the `NonlinearFit` package for obtaining reasonably good start values.

■ 5. Discussion of the Results Using *Mathematica* 5.1

Darren Glosemeyer

Kernel Developer
Kernel Technology Group
Wolfram Research, Inc.
darreng@wolfram.com

The following results were obtained in *Mathematica* 5.1 on an Intel® Pentium 4 Windows XP computer. Version 5.1 does a little better on some examples than 5.0.

```
In[1]:=  $Version

Out[1]=  5.1 for Microsoft Windows (October 25, 2004)
```

The following comments are for the examples for which problems were noted in 5.0. Results for the other examples in the NIST benchmarks still give correct results in 5.1. For most examples, just the parameter tables are shown; however, the s^2 values were also checked against the benchmark.

For three examples, differences between *Mathematica*'s results and the benchmark result are attributable to numerical error in machine-precision arithmetic, numerical error in the NIST data, or a combination of those two sources. In the other examples, `NonlinearRegress` either gets a correct result with default settings or obtains a correct result with an additional option in 5.1.

```
In[2]:= << Statistics`
```

Wampler1

The numerical error in this example is a result of numerical error in machine-precision computation and can be expected.

```
In[3]:= w1 = {{0, 1}, {1, 6}, {2, 63}, {3, 364},
          {4, 1365}, {5, 3906}, {6, 9331}, {7, 19608}, {8, 37449},
          {9, 66430}, {10, 111111}, {11, 177156}, {12, 271453},
          {13, 402234}, {14, 579195}, {15, 813616}, {16, 1118481},
          {17, 1508598}, {18, 2000719}, {19, 2613660}, {20, 3368421}};
```

```
In[4]:= ParameterTable /. Regress[w1, {1, x, x^2, x^3, x^4, x^5},
          x, RegressionReport → {ParameterTable}]
```

```
Out[4]//TableForm=
              Estimate      SE                  TStat               PValue
       1      1.            4.98764×10^-10      2.00496×10^9        0.
       x      1.            5.47711×10^-10      1.82578×10^9        0.
       x^2    1.            1.80599×10^-10      5.53713×10^9        0.
       x^3    1.            2.35152×10^-11      4.25258×10^10       0.
       x^4    1.            1.30828×10^-12      7.64361×10^11       0.
       x^5    1.            2.60293×10^-14      3.84182×10^13       0.
```

Significance arithmetic (variable-precision arithmetic) can be used to get results of a desired precision or accuracy. `Regress` can be forced to use significance arithmetic by applying `SetPrecision` to the data. Here is an example with precision 20 data with the result displayed at machine-precision. Because there is no random error in the original data, division by 0 will result in error messages when attempting to compute the t statistics and p-values.

```
In[5]:= Off[Power::"infy", Less::"nord"];
        N[ParameterTable /. Regress[SetPrecision[w1, 20],
           {1, x, x^2, x^3, x^4, x^5}, x, RegressionReport → {ParameterTable}]]
```

```
Out[6]//TableForm=
            Estimate     SE      TStat               PValue
       1.   1.           0.      ComplexInfinity     Indeterminate
       x    1.           0.      ComplexInfinity     Indeterminate
       x^2  1.           0.      ComplexInfinity     Indeterminate
       x^3  1.           0.      ComplexInfinity     Indeterminate
       x^4  1.           0.      ComplexInfinity     Indeterminate
       x^5  1.           0.      ComplexInfinity     Indeterminate
```

□ Wampler2

In this example, there are two sources of numerical error: rounding in the original data and machine-precision arithmetic. While the NIST data was generated from a polynomial expression, the data is only given to five digits of accuracy, resulting in some roundoff or truncation error in the data.

```
In[7]:= w2 = {{0, 1.}, {1, 1.11111}, {2, 1.24992}, {3, 1.42753}, {4, 1.65984},
           {5, 1.96875}, {6, 2.38336}, {7, 2.94117}, {8, 3.68928},
           {9, 4.68559}, {10, 6.}, {11, 7.71561}, {12, 9.92992},
           {13, 12.75603}, {14, 16.32384}, {15, 20.78125}, {16, 26.29536},
           {17, 33.05367}, {18, 41.26528}, {19, 51.16209}, {20, 63.}};
```

```
In[8]:= ParameterTable /. Regress[w2, {1, x, x^2, x^3, x^4, x^5},
          x, RegressionReport → {ParameterTable}]
```

```
Out[8]//TableForm=
            Estimate     SE                TStat             PValue
       1    1.           1.52366×10^-14    6.56316×10^13     0.
       x    0.1          1.67318×10^-14    5.97663×10^12     0.
       x^2  0.01         5.51706×10^-15    1.81256×10^12     0.
       x^3  0.001        7.18357×10^-16    1.39207×10^12     0.
       x^4  0.0001       3.99663×10^-17    2.50211×10^12     0.
       x^5  0.00001      7.9516×10^-19     1.25761×10^13     0.
```

As in the case of Wampler1, more precise results can be obtained by using higher precision input for the regression.

```
In[9]:= Off[Power::"infy", Less::"nord"];
        N[ParameterTable /. Regress[SetPrecision[w2, 17],
           {1, x, x^2, x^3, x^4, x^5}, x, RegressionReport → {ParameterTable}]]
```

```
Out[10]//TableForm=
              Estimate    SE      TStat              PValue
        1.    1.          0.      ComplexInfinity    Indeterminate
        x     0.1         0.      ComplexInfinity    Indeterminate
        x^2   0.01        0.      ComplexInfinity    Indeterminate
        x^3   0.001       0.      ComplexInfinity    Indeterminate
        x^4   0.0001      0.      ComplexInfinity    Indeterminate
        x^5   0.00001     0.      ComplexInfinity    Indeterminate
```

However, increasing the precision of the input cannot make up for the fact that there is some numerical error in the original data. As the precision is increased, the significance of the roundoff or truncation error in the data from the NIST will show up as small deviations from the benchmark results.

```
In[11]:= Off[Power::"infy", Less::"nord"];
         N[ParameterTable /. Regress[SetPrecision[w2, 30],
            {1, x, x^2, x^3, x^4, x^5}, x, RegressionReport → {ParameterTable}]]
```

```
Out[12]//TableForm=
              Estimate    SE                 TStat              PValue
        1.    1.          6.38509 × 10^-16   1.56615 × 10^15    1.60271 × 10^-220
        x     0.1         7.01171 × 10^-16   1.42618 × 10^14    6.52694 × 10^-205
        x^2   0.01        2.312 × 10^-16     4.32526 × 10^13    3.86516 × 10^-197
        x^3   0.001       3.01037 × 10^-17   3.32185 × 10^13    2.02606 × 10^-195
        x^4   0.0001      1.67484 × 10^-18   5.97071 × 10^13    3.06874 × 10^-199
        x^5   0.00001     3.33223 × 10^-20   3.00099 × 10^14    9.2976 × 10^-210
```

□ MGH17

Each value in the first set of starting points is about two orders of magnitude from the optimal value. Overflows are quickly encountered using the default Levenberg–Marquardt method. Despite the tough starting values, good results can be obtained using Newton's method instead of the default Levenberg–Marquardt method.

```
In[13]:= mgh17 = {{0., 0.844}, {10., 0.908}, {20., 0.932}, {30., 0.936},
            {40., 0.925}, {50., 0.908}, {60., 0.881}, {70., 0.85}, {80., 0.818},
            {90., 0.784}, {100., 0.751}, {110., 0.718}, {120., 0.685},
            {130., 0.658}, {140., 0.628}, {150., 0.603}, {160., 0.58},
            {170., 0.558}, {180., 0.538}, {190., 0.522}, {200., 0.506},
            {210., 0.49}, {220., 0.478}, {230., 0.467}, {240., 0.457},
            {250., 0.448}, {260., 0.438}, {270., 0.431}, {280., 0.424},
            {290., 0.42}, {300., 0.414}, {310., 0.411}, {320., 0.406}};
```

```
In[14]:= ParameterCITable /. NonlinearRegress[mgh17, β1 + β2 ⅇ^(-β4 x) + β3 ⅇ^(-β5 x),
           x, {{β1, 50}, {β2, 150}, {β3, -100}, {β4, 1}, {β5, 2}},
           RegressionReport → {ParameterCITable}, Method → Newton]
```

```
Out[14]//TableForm=
          Estimate      Asymptotic SE    CI
     β1   0.37541       0.00207232       {0.371165, 0.379655}
     β2   1.93585       0.220317         {1.48455, 2.38715}
     β3   -1.46469      0.221757         {-1.91894, -1.01044}
     β4   0.0128675     0.000448614      {0.0119486, 0.0137865}
     β5   0.0221227     0.00089472       {0.0202899, 0.0239555}
```

As in 5.0, this example converges to the correct result from the second set of starting values with default options.

□ Lanczos1

With both sets of starting points, this example converges to the expected parameter estimates with default options in 5.1. The SEs and s^2 values are on the same order as the benchmark values, which is reasonable given the magnitudes of the SEs and s^2 and the fact that the benchmark data is given to 13 digits. For the first starting points, the SEs agree with the benchmark in at least the first digit. For the second starting points, the SEs agree with the benchmark in at least the first two digits.

```
In[15]:= l1 = {{0., 2.5134}, {0.05, 2.044333373291},
             {0.1, 1.668404436564}, {0.15, 1.366418021208},
             {0.2, 1.123232487372}, {0.25, 0.9268897180037},
             {0.3, 0.7679338563728}, {0.35, 0.6388775523106},
             {0.4, 0.5337835317402}, {0.45, 0.4479363617347},
             {0.5, 0.377584788435}, {0.55, 0.3197393199326},
             {0.6, 0.2720130773746}, {0.65, 0.2324965529032},
             {0.7, 0.1996589546065}, {0.75, 0.1722704126914},
             {0.8, 0.1493405660168}, {0.85, 0.1300700206922},
             {0.9, 0.1138119324644}, {0.95, 0.1000415587559},
             {1., 0.0883320908454}, {1.05, 0.0783354401935},
             {1.1, 0.06976693743449}, {1.15, 0.06239312536719}};
```

```
In[16]:= res = NonlinearRegress[l1, β1 e^(-β2 x) + β3 e^(-β4 x) + β5 e^(-β6 x), x,
            {{β1, 1.2}, {β2, .3}, {β3, 5.6}, {β4, 5.5}, {β5, 6.5}, {β6, 7.6}},
            RegressionReport → {ParameterCITable, EstimatedVariance}];
         ParameterCITable /. res
         EstimatedVariance /. res

Out[17]//TableForm=
              Estimate     Asymptotic SE          CI
         β1   0.0951       5.91894 × 10^-11       {0.0951, 0.0951}
         β2   1.           3.04816 × 10^-10       {1., 1.}
         β3   0.8607       1.50628 × 10^-10       {0.8607, 0.8607}
         β4   3.           3.69559 × 10^-10       {3., 3.}
         β5   1.5576       2.08763 × 10^-10       {1.5576, 1.5576}
         β6   5.           1.22684 × 10^-10       {5., 5.}

Out[18]= 9.78511 × 10^-27

In[19]:= res = NonlinearRegress[l1, β1 e^(-β2 x) + β3 e^(-β4 x) + β5 e^(-β6 x), x,
            {{β1, .5}, {β2, .7}, {β3, 3.6}, {β4, 4.2}, {β5, 4}, {β6, 6.3}},
            RegressionReport → {ParameterCITable, EstimatedVariance}];
         ParameterCITable /. res
         EstimatedVariance /. res

Out[20]//TableForm=
              Estimate     Asymptotic SE          CI
         β1   0.0951       5.35261 × 10^-11       {0.0951, 0.0951}
         β2   1.           2.75651 × 10^-10       {1., 1.}
         β3   0.8607       1.36216 × 10^-10       {0.8607, 0.8607}
         β4   3.           3.34199 × 10^-10       {3., 3.}
         β5   1.5576       1.88788 × 10^-10       {1.5576, 1.5576}
         β6   5.           1.10946 × 10^-10       {5., 5.}

Out[21]= 8.00218 × 10^-27
```

With an increased `AccuracyGoal`, s^2 values matching the benchmark to three digits can be obtained.

```
In[22]:= EstimatedVariance /. NonlinearRegress[l1, β1 e^(-β2 x) + β3 e^(-β4 x) + β5 e^(-β6 x), x,
           {{β1, 1.2}, {β2, .3}, {β3, 5.6}, {β4, 5.5}, {β5, 6.5}, {β6, 7.6}},
           RegressionReport → {EstimatedVariance}, AccuracyGoal → 15]

Out[22]= 7.94334 × 10^-27

In[23]:= EstimatedVariance /. NonlinearRegress[l1, β1 e^(-β2 x) + β3 e^(-β4 x) + β5 e^(-β6 x), x,
           {{β1, .5}, {β2, .7}, {β3, 3.6}, {β4, 4.2}, {β5, 4}, {β6, 6.3}},
           RegressionReport → {EstimatedVariance}, AccuracyGoal → 15]

Out[23]= 7.94795 × 10^-27
```

□ Lanczos2

With both sets of starting points, this example converges to the expected result with default options in 5.1.

```
In[24]:= l2 = {{0., 2.5134}, {0.05, 2.04433}, {0.1, 1.6684}, {0.15, 1.36642},
            {0.2, 1.12323}, {0.25, 0.92689}, {0.3, 0.767934}, {0.35, 0.638878},
            {0.4, 0.533784}, {0.45, 0.447936}, {0.5, 0.377585},
            {0.55, 0.319739}, {0.6, 0.272013}, {0.65, 0.232497},
            {0.7, 0.199659}, {0.75, 0.17227}, {0.8, 0.149341}, {0.85, 0.13007},
            {0.9, 0.113812}, {0.95, 0.100042}, {1., 0.0883321},
            {1.05, 0.0783354}, {1.1, 0.0697669}, {1.15, 0.0623931}};
```

```
In[25]:= ParameterCITable /. NonlinearRegress[l2, β1 e^(-β2 x) + β3 e^(-β4 x) + β5 e^(-β6 x), x,
           {{β1, 1.2}, {β2, .3}, {β3, 5.6}, {β4, 5.5}, {β5, 6.5}, {β6, 7.6}},
           RegressionReport → {ParameterCITable}]
```

```
Out[25]//TableForm=
              Estimate      Asymptotic SE     CI
        β1    0.096251      0.000667706       {0.0948482, 0.0976538}
        β2    1.00573       0.00339896        {0.998592, 1.01287}
        β3    0.864247      0.00171858        {0.860636, 0.867858}
        β4    3.00783       0.0041707         {2.99907, 3.01659}
        β5    1.5529        0.00237444        {1.54791, 1.55789}
        β6    5.00288       0.00139588        {4.99995, 5.00581}
```

```
In[26]:= ParameterCITable /. NonlinearRegress[l2, β1 e^(-β2 x) + β3 e^(-β4 x) + β5 e^(-β6 x), x,
           {{β1, .5}, {β2, .7}, {β3, 3.6}, {β4, 4.2}, {β5, 4}, {β6, 6.3}},
           RegressionReport → {ParameterCITable}]
```

```
Out[26]//TableForm=
              Estimate      Asymptotic SE     CI
        β1    0.096251      0.000667706       {0.0948482, 0.0976538}
        β2    1.00573       0.00339896        {0.998592, 1.01287}
        β3    0.864247      0.00171858        {0.860636, 0.867858}
        β4    3.00783       0.0041707         {2.99907, 3.01659}
        β5    1.5529        0.00237444        {1.54791, 1.55789}
        β6    5.00288       0.00139588        {4.99995, 5.00581}
```

□ MGH09

This example converges a little faster in 5.1 than in 5.0. Convergence is obtained for the first starting points with `MaxIterations→400`, as opposed to `MaxIterations→600` in 5.0.

```
In[27]:= mgh09 = {{4., 0.1957}, {2., 0.1947}, {1., 0.1735}, {0.5, 0.16},
            {0.25, 0.0844}, {0.167, 0.0627}, {0.125, 0.0456}, {0.1, 0.0342},
            {0.0833, 0.0323}, {0.0714, 0.0235}, {0.0625, 0.0246}};
```

```
In[28]:= ParameterCITable /. NonlinearRegress[mgh09,
```
$$\frac{\beta_1\ (x^2 + x\ \beta_2)}{x^2 + x\ \beta_3 + \beta_4}$$
```
         , x, {{β1, 25}, {β2, 39}, {β3, 41.5}, {β4, 39}},
         RegressionReport → {ParameterCITable}, MaxIterations → 400]

Out[28]//TableForm=
        Estimate     Asymptotic SE    CI
  β1    0.192807     0.0114353        {0.165767, 0.219847}
  β2    0.191282     0.196332         {-0.27297, 0.655534}
  β3    0.123057     0.080842         {-0.0681045, 0.314218}
  β4    0.136062     0.0900255        {-0.0768142, 0.348939}
```

Correct results are obtained from the second set of starting points with the default option settings, as is the case in 5.0.

□ MGH10

Convergence from each set of starting values was obtained with `MaxIterations`→300 in 5.1. This is an improvement for the first set of points, for which `MaxIterations`→500 was used to obtain the benchmark result in 5.0.

```
In[29]:= mgh10 = {{50., 34780.}, {55., 28610.}, {60., 23650.}, {65., 19630.},
           {70., 16370.}, {75., 13720.}, {80., 11540.}, {85., 9744.},
           {90., 8261.}, {95., 7030.}, {100., 6005.}, {105., 5147.},
           {110., 4427.}, {115., 3820.}, {120., 3307.}, {125., 2872.}};

In[30]:= ParameterCITable /. NonlinearRegress[
```
$$\text{mgh10},\ \beta_1\, e^{\frac{\beta_2}{x+\beta_3}}$$
```
         , x, {{β1, 2}, {β2, 400000}, {β3, 25000}},
         RegressionReport → {ParameterCITable}, MaxIterations → 300]

Out[30]//TableForm=
        Estimate       Asymptotic SE    CI
  β1    0.00560964     0.000156879      {0.00527072, 0.00594855}
  β2    6181.35        23.309           {6130.99, 6231.7}
  β3    345.224        0.784861         {343.528, 346.919}

In[31]:= ParameterCITable /.
        NonlinearRegress[mgh10,
```
$$\beta_1\, e^{\frac{\beta_2}{x+\beta_3}}$$
```
         , x, {{β1, .02}, {β2, 4000}, {β3, 250}},
         RegressionReport → {ParameterCITable}, MaxIterations → 300]

Out[31]//TableForm=
        Estimate       Asymptotic SE    CI
  β1    0.00560964     0.000156879      {0.00527072, 0.00594855}
  β2    6181.35        23.309           {6130.99, 6231.7}
  β3    345.224        0.784861         {343.528, 346.919}
```

□ Bennett5

For this example, convergence in 5.1 can be obtained with half the iterations used in 5.0. From the first set of starting values, correct results are obtained with `MaxIterations→750`. From the second set of starting values, setting `MaxItera⋱`
`tions→200` is sufficient to obtain the benchmark results.

```
In[32]:= bennett5 = {{7.447168, -34.834702}, {8.102586, -34.3932},
           {8.452547, -34.152901}, {8.711278, -33.979099},
           {8.916774, -33.845901}, {9.087155, -33.732899}, {9.23259, -33.640301},
           {9.359535, -33.5592}, {9.472166, -33.486801}, {9.573384, -33.4231},
           {9.665293, -33.365101}, {9.749461, -33.313}, {9.827092, -33.260899},
           {9.899128, -33.2174}, {9.966321, -33.176899}, {10.02928, -33.139198},
           {10.08851, -33.101601}, {10.14443, -33.066799}, {10.19738, -33.035},
           {10.24767, -33.003101}, {10.29556, -32.971298}, {10.34125, -32.942299},
           {10.38495, -32.916302}, {10.42682, -32.890202}, {10.467, -32.864101},
           {10.50564, -32.841}, {10.54283, -32.817799}, {10.57869, -32.797501},
           {10.61331, -32.7743}, {10.64678, -32.757}, {10.67915, -32.733799},
           {10.71052, -32.7164}, {10.74092, -32.6991}, {10.77044, -32.678799},
           {10.7991, -32.6614}, {10.82697, -32.644001}, {10.85408, -32.626701},
           {10.88047, -32.612202}, {10.90619, -32.597698}, {10.93126, -32.583199},
           {10.95572, -32.568699}, {10.97959, -32.554298}, {11.00291, -32.539799},
           {11.0257, -32.525299}, {11.04798, -32.510799}, {11.06977, -32.499199},
           {11.0911, -32.487598}, {11.11198, -32.473202}, {11.13244, -32.461601},
           {11.15248, -32.435501}, {11.17213, -32.435501}, {11.19141, -32.4268},
           {11.21031, -32.4123}, {11.22887, -32.400799}, {11.24709, -32.392101},
           {11.26498, -32.380501}, {11.28256, -32.366001}, {11.29984, -32.3573},
           {11.31682, -32.348598}, {11.33352, -32.339901}, {11.34994, -32.3284},
           {11.3661, -32.319698}, {11.382, -32.311001}, {11.39766, -32.2994},
           {11.41307, -32.290699}, {11.42824, -32.282001}, {11.4432, -32.2733},
           {11.45793, -32.264599}, {11.47244, -32.256001}, {11.48675, -32.247299},
           {11.50086, -32.238602}, {11.51477, -32.2299}, {11.52849, -32.224098},
           {11.54202, -32.215401}, {11.55538, -32.2038}, {11.56855, -32.198002},
           {11.58156, -32.1894}, {11.59442, -32.183601}, {11.607121, -32.1749},
           {11.61964, -32.169102}, {11.632, -32.1633}, {11.64421, -32.154598},
           {11.65628, -32.145901}, {11.6682, -32.140099}, {11.67998, -32.131401},
           {11.69162, -32.125599}, {11.70313, -32.119801}, {11.71451, -32.111198},
           {11.72576, -32.1054}, {11.73688, -32.096699}, {11.74789, -32.0909},
           {11.75878, -32.088001}, {11.76955, -32.0793}, {11.7802, -32.073502},
           {11.79073, -32.067699}, {11.80116, -32.061901}, {11.81148, -32.056099},
           {11.8217, -32.050301}, {11.83181, -32.044498}, {11.84182, -32.038799},
           {11.85173, -32.033001}, {11.86155, -32.027199}, {11.87127, -32.0243},
           {11.88089, -32.018501}, {11.89042, -32.012699}, {11.89987, -32.004002},
           {11.90922, -32.001099}, {11.91849, -31.9953}, {11.92768, -31.9895},
           {11.93678, -31.9837}, {11.94579, -31.9779}, {11.95473, -31.972099},
           {11.96359, -31.969299}, {11.97237, -31.963501}, {11.98107, -31.957701},
           {11.9897, -31.9519}, {11.99826, -31.9461}, {12.00674, -31.9403},
           {12.01515, -31.937401}, {12.02349, -31.931601}, {12.03176, -31.9258},
           {12.03997, -31.922899}, {12.0481, -31.917101}, {12.05617, -31.911301},
           {12.06418, -31.9084}, {12.07212, -31.902599}, {12.08001, -31.8969},
           {12.08782, -31.893999}, {12.09558, -31.888201}, {12.10328, -31.8853},
           {12.11092, -31.882401}, {12.1185, -31.8766}, {12.12603, -31.873699},
           {12.1335, -31.867901}, {12.14091, -31.862101}, {12.14827, -31.8592},
           {12.15557, -31.8563}, {12.16283, -31.8505}, {12.17003, -31.8447},
           {12.17717, -31.841801}, {12.18427, -31.8389}, {12.19132, -31.833099},
           {12.19832, -31.8302}, {12.20527, -31.827299}, {12.21217, -31.8216},
           {12.21903, -31.818701}, {12.22584, -31.812901}, {12.2326, -31.809999},
           {12.23932, -31.8071}, {12.24599, -31.8013}, {12.25262, -31.798401},
           {12.2592, -31.7955}, {12.26575, -31.7897}, {12.27224, -31.7868}};
```

```
In[33]:= ParameterCITable /. NonlinearRegress[bennett5,
           β1 (x + β2)^(-1/β3), x, {{β1, -2000}, {β2, 50}, {β3, .8}},
           RegressionReport → {ParameterCITable}, MaxIterations → 750]

Out[33]//TableForm=
              Estimate      Asymptotic SE    CI
      β1      -2523.51      297.152          {-3110.62, -1936.39}
      β2      46.7366       1.24489          {44.2769, 49.1962}
      β3      0.932185      0.0202723        {0.892131, 0.972239}
```

```
In[34]:= ParameterCITable /. NonlinearRegress[bennett5,
           β1 (x + β2)^(-1/β3), x, {{β1, -1500}, {β2, 45}, {β3, .85}},
           RegressionReport → {ParameterCITable}, MaxIterations → 200]

Out[34]//TableForm=
              Estimate      Asymptotic SE    CI
      β1      -2523.51      297.152          {-3110.62, -1936.39}
      β2      46.7366       1.24489          {44.2769, 49.1962}
      β3      0.932185      0.0202723        {0.892131, 0.972239}
```

■ Acknowledgments

The research on which this article is based has been supported by the Maryland Agricultural Experiment Station. I am also indebted to two anonymous referees, as well as Daniel Lichtblau, Bruce McCullough, Charles Renfro, and, above all, Colin Rose for helpful comments. The author assumes responsibility for remaining errors.

■ Endnotes

(1) www.nag.co.uk/stats/ae_soft.asp

(2) www.itl.nist.gov/div898/strd

(3) McCullough [48] has also looked at several popular econometrics packages, such as E-Views, LIMDEP, SHAZAM, and TSP. The results are generally comparable to those obtained for SAS.

■ References

[1] B. D. McCullough and H. D. Vinod, "The Numerical Reliability of Econometric Software," *Journal of Economic Literature,* **37**, 1999 pp. 633–665.

[2] B. D. McCullough and B. Wilson, "On the Accuracy of Statistical Procedures in Microsoft Excel 97," *Computational Statistics and Data Analysis,* **31**, 1999 pp. 27–37.

[3] B. D. McCullough, "The Accuracy of *Mathematica* 4 as a Statistical Package," *Computational Statistics,* **15**, 2000 pp. 279–299.

[4] H. D. Vinod, "Review of GAUSS for Windows, Including Its Numerical Accuracy," *Journal of Applied Econometrics,* **15**, 2000 pp. 211–220.

[5] B. D. McCullough, "Assessing the Reliability of Statistical Software: Parts I and II," *The American Statistician*, **52**, 1998 pp. 358–366; **53**, 1999 pp. 149–159.

[6] J. Norris, Calibration of Ozone Monitors, NIST, unpublished.

[7] P. Pontius, Load Cell Calibration, NIST, unpublished.

[8] K. Eberhardt, NoInt1 and NoInt2, NIST, unpublished.

[9] A. Filippelli, Filip, NIST, unpublished.

[10] J. W. Longley, "An Appraisal of Least Squares Programs for the Electronic Computer from the Viewpoint of the User," *Journal of the American Statistical Association*, **62**, 1967 pp. 819–841.

[11] R. H. Wampler, "A Report of the Accuracy of Some Widely-Used Least Squares Computer Programs," *Journal of the American Statistical Association*, **65**, 1970 pp. 549–565.

[12] D. Misra, Dental Research Monomolecular Adsorption Study, NIST, unpublished, 1978.

[13] D. Chwirut, Ultrasonic Reference Block Study, NIST, unpublished, 1979.

[14] C. Lanczos, *Applied Analysis*, Englewood Cliffs, NJ: Prentice Hall, 1956 pp. 272–280.

[15] B. Rust, Gauss 1, 2, and 3, NIST, unpublished, 1996.

[16] C. Daniel and F. S. Wood, *Fitting Equations to Data*, 2nd ed., New York: John Wiley & Sons, 1980 pp. 428–431.

[17] R. Kirby, Scanning Electron Microscope Line Width Standards, NIST, unpublished, 1979.

[18] T. Hahn, Copper Thermal Expansion Study, NIST, unpublished, 1979.

[19] W. Nelson, "Analysis of Performance-Degradation Data," *IEEE Transactions on Reliability,* **2, R-30** (2), 1981 pp. 149–155.

[20] M. R. Osborne, "Some Aspects of Nonlinear Least Squares Calculations," *Numerical Methods for Nonlinear Optimization* (F. A. Lootsma, ed.), New York: Academic Press, 1972 pp. 171–189.

[21] L. Roszman, Quantum Defects for Sulfur I Atom, NIST, unpublished, 1979.

[22] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Englewood Cliffs, NJ: Prentice Hall, 1989 pp. 441–445.

[23] J. S. Kowalik and M. R. Osborne, *Methods for Unconstrained Optimization Problems*, New York: Elsevier North-Holland, 1978.

[24] R. Thurber, Semiconductor Electron Mobility Modeling, NIST, unpublished, 1979.

[25] G. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters,* New York: John Wiley & Sons, 1978 pp. 483–487.

[26] D. A. Ratkowsky, *Nonlinear Regression Modelling: A Unified Practical Approach,* New York: Marcel Dekker, 1983 pp. 61–62 and 88.

[27] R. R. Meyer, "Theoretical and Computational Aspects of Nonlinear Regression," *Nonlinear Programming* (J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds.), New York: Academic Press, 1970 pp. 465–486.

[28] K. Eckerle, Circular Interference Transmittance Study, NIST, unpublished, 1979.

[29] L. Bennett, L. Swartzendruber, and H. Brown, Bennett5, NIST, unpublished, 1994.

[30] K. L. Hiebert, "An Evaluation of Mathematical Software that Solves Nonlinear Least Squares Problems: Part I," *ACM Transactions on Mathematical Software*, **7**, 1981 pp. 1–16.

[31] S. Wolfram, *The Mathematica Book,* 5th ed., Champaign, Oxford: Wolfram Media/ Cambridge University Press, 2003.

[32] R. Davidson and J. G. MacKinnon, "Nonlinear Regression Models and Nonlinear Least Squares," *Estimation and Inference in Econometrics,* New York: Oxford University Press, 1993 pp. 41–65.

[33] W. H. Greene, "Nonlinear Regression Models," *Econometric Analysis*, 4th ed., Englewood Cliffs, NJ: Prentice Hall, 2000 pp. 416–454.

[34] W. Haerdle, *Applied Nonparametric Regression*, New York: Cambridge University Press, 1990.

[35] A. R. Pagan and A. Ullah, *Nonparametric Econometrics,* New York: Cambridge University Press, 1998.

[36] A. R. Gallant, *Nonlinear Statistical Models,* New York: John Wiley & Sons, 1987.

[37] D. M. Bates and D. G. Watts, *Nonlinear Regression Analysis and Its Applications,* New York, John Wiley & Sons, 1988.

[38] G. A. F. Seber and C. J. Wild, *Nonlinear Regression,* New York: John Wiley & Sons, 1989.

[39] A. R. Gallant, "Nonlinear Regression," *The American Statistician*, **29**, 1975 pp. 73–81.

[40] H. O. Hartley, "The Modified Gauss-Newton Method for the Fitting of Nonlinear Regression Functions by Least Squares," *Technometrics*, **3**, 1961 pp. 269–280.

[41] H. O. Hartley and A. Booker, "Non-linear Least Squares Estimation," *The Annals of Mathematical Statistics*, **36**, 1965 pp. 638–650.

[42] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, **11**, 1963 pp. 431–441.

[43] K. Levenberg, "A Method for the Solution of Certain Nonlinear Problems in Least Squares," *Quarterly of Applied Mathematics*, **2**, 1944 pp. 164–168.

[44] K. L. Hiebert, "An Evaluation of Mathematical Software That Solves Nonlinear Least Squares Problems: Part II," *ACM Transactions on Mathematical Software*, **8**, 1982 pp. 5–20.

[45] R. I. Jennrich, "Asymptotic Properties of Nonlinear Least Squares Estimators," *Annals of Mathematical Statistics*, **40**, 1969 pp. 633–643.

[46] J. J. More, B. S. Garbow, and K. E. Hillstrom, "Testing Unconstrained Optimization Software," *ACM Transactions on Mathematical Software*, **7**, 1981 pp. 17–41.

[47] R. D. Skeel and J. B. Keiper, *Elementary Numerical Computing with Mathematica*, New York: McGraw-Hill, 1993.

[48] B. D. McCullough, "The Reliability of Econometric Software: E-Views, LIMDEP, SHAZAM, and TSP," *Journal of Applied Econometrics*, **14**, 1999 pp. 191–202.

■ Additional Material

AppendicesAccuracy.nb

Available at www.mathematica-journal.com/issue/v9i4/download.

About the Author

Marc Nerlove is a professor of economics at the University of Maryland, College Park, MD. He received his B.A. in mathematics from the University of Chicago and his M.A. and Ph.D. degrees in economics from Johns Hopkins University. His research interests include environmental and resource economics, likelihood inference in econometrics, and agricultural development in the context of two-sector economic growth.

Marc Nerlove
Department of Agricultural and Resource Economics
University of Maryland
College Park, MD 20742
mnerlove@arec.umd.edu
www.arec.umd.edu/mnerlove/mnerlove.htm