

8

Some Details of Nonlinear Estimation

B. D. McCullough

8.1 INTRODUCTION

The traditional view adopted by statistics and econometrics texts is that, in order to solve a nonlinear least squares (NLS) or nonlinear maximum likelihood (NML) problem, a researcher need only use a computer program. This point of view maintains two implicit assumptions: (1) one program is as good as another; and (2) the output from a computer is reliable. Both of these maintained assumptions are false.

Recently the National Institute of Standards and Technology (NIST) released the “Statistical Reference Datasets” (StRD), a collection of numerical accuracy benchmarks for statistical software. It has four suites of tests: univariate summary statistics, one-way analysis of variance, linear regression, and nonlinear least squares. Within each suite the problems are graded according to level of difficulty: low, average, and high. NIST provides certified answers to fifteen digits for linear problems, and eleven digits for nonlinear problems. Complete details can be found at <http://www.nist.gov/itl/div898/strd>.

Nonlinear problems require starting values, and all the StRD nonlinear problems come with two sets of starting values. Start I is far from the solution and makes the problem hard to solve. Start II is near to the solution and makes the problem easy to solve. Since the purpose of benchmark testing is to say something useful about

Thanks to P. Spelluci and J. Nocedal for useful discussions, and to T. Harrison for comments.

the underlying algorithm, Start I is more important, since the solver is more likely to report false convergence from Start I than from Start II. To see this, consider two possible results: (A) that the solver can correctly solve an easy problem; and (B) that the solver can stop at a point that is not a solution and nonetheless declare that it has found a solution. Clearly, (B) constitutes more useful information than (A).

The StRD nonlinear suite has been applied to numerous statistical and econometric software packages including SAS, SPSS, S-Plus, Gauss, TSP, LIMDEP, SHAZAM, EVIEWS, and several others. Some packages perform very well on these tests, while others exhibit a marked tendency to report false convergence, *i.e.*, the nonlinear solver stops at a point that is not a minimum and nevertheless reports that it has found a minimum. Thus, the first point of the traditional view is shown to be false: some packages are very good at solving nonlinear problems, and other packages are very bad. Even the packages that perform well on the nonlinear suite of tests can return false convergence if not used carefully, and so the second point of the traditional view is shown to be false. Users of nonlinear solvers need some way to protect themselves against false results. The present chapter offers useful guidance on the matter.

Section 2 presents the basic ideas behind a nonlinear solver. Section 3 considers some details on the mechanics of nonlinear solvers. Section 4 analyzes a simple example where a nonlinear solver produces several incorrect answers to the same problem. Section 4 offers a list of ways that a user can guard against incorrect answers. Section 5 offers Monte Carlo evidence on profile likelihood. Section 6 offers conclusions, including what to look for in a nonlinear solver.

8.2 AN OVERVIEW OF ALGORITHMS

Gradient-based methods for finding the set of coefficients $\hat{\beta}$ that minimize a nonlinear least squares function¹ take the form on the k -th iteration

$$\hat{\beta}^{k+1} = \hat{\beta}^k + \lambda_k d_k \quad (8.1)$$

where $\beta = [\beta_1, \beta_2, \dots, \beta_m]'$ is the vector of parameters to be estimated. The objective function is the sum of squared residuals, denoted $S(\hat{\beta})$. From some selected vector of starting values, β^0 , the iterative process proceeds by taking a step of length λ in some direction d . Different choices for λ and d give rise to different algorithms. The so-called *gradient methods* are based on $d = Wg$ where g is the gradient of the objective function and W is some matrix.

One approach to choosing λ and d is to take a linear approximation to the objective function

$$F(\hat{\beta}_k + \lambda) \equiv F_k + g'_k \lambda \quad (8.2)$$

¹The remainder of this section addresses nonlinear least squares directly, but much of the discussion applies, *mutatis mutandis*, to nonlinear maximum likelihood.

This leads to the choice $W = I$ where I is the identity matrix, and yields $d = -g_k$, thus producing the algorithm known as *steepest descent*. While it requires numerical evaluation of only the function and the gradient, it makes no use of the curvature of the function, *i.e.*, it makes no use of the Hessian. Thus the steepest descent method has the disadvantage that it is very slow: the steepest descent method can require hundreds of iterations to do what other algorithms can do in just several iterations.

Another approach to choosing λ and d is to take a quadratic approximation to the objective function. Then a first-order Taylor expansion about the current iterate yields

$$F(\beta_k + d_k) \approx F(\beta_k) + g'_k d_k + \frac{1}{2} d'_k H_k d_k \quad (8.3)$$

where g_k and H_k are the gradient and Hessian, respectively, at the k -th iteration. An associated quadratic function in d_k can be defined

$$h(\lambda) = g'_k d_k + \frac{1}{2} d'_k H_k d_k \quad (8.4)$$

A stationary point of the quadratic function, d_k , will satisfy

$$H_k d_k = -g_k \quad (8.5)$$

When the direction d_k is a solution of the system of equations given by Eq. 8.5, then d_k is the Newton direction. If, further, the step length is unity, then the method is called Newton's method. If the step-length is other than unity, then the method is a modified Newton method. The Newton method is very powerful because it makes full use of the curvature information. However, it has three primary defects, two of which are remediable.

The first remediable problem is that the Newton step, $\lambda = 1$, is not always a good choice. One reason for choosing λ to be other than unity is because if $\lambda \equiv 1$, then $\hat{\beta}^{k+1}$ is not guaranteed to be closer to a solution than $\hat{\beta}^k$. One way around this is to project a ray from $\hat{\beta}^k$ and then search along this ray for an acceptable value of λ ; this is called a *line search*. In particular, solving

$$\min_{\lambda > 0} f(\hat{\beta}^k + \lambda d_k) \quad (8.6)$$

produces an *exact line search*. Because this can be computationally expensive, frequently an algorithm will find a value λ that roughly approximates the minimum; this is called an *inexact line search*. Proofs for some theorems on the convergences of various methods require that the line search be exact.

The second problem is that computing first derivatives for a method that only uses gradient information is much less onerous than also computing second derivatives for a method that explicitly uses Hessian information, too. Analytic derivatives are well-known to be more reliable than their numerical counterparts. However, frequently it is the case that the user must calculate and then code these derivatives, a not insubstantial undertaking. When the user must calculate and code derivatives, frequently the user instead relies solely on numerical derivatives (Dennis, 1984, p.1766).Some

packages offer automatic differentiation, in which a specialized subroutine automatically calculates analytic derivatives, thus easing appreciably the burden on the user. See Nocedal and Wright (1999, Chapter 7) for a discussion of automatic derivatives. Automatic differentiation is not perfect, and in rare occasions the automatic derivatives are not numerically efficient. In such a case, it may be necessary to rewrite or otherwise simplify the expressions for the derivatives. Of course, it is also true that user-supplied analytic derivatives may need rewriting or simplification.

The third and irremediable problem with the Newton method is that for points far from the solution, the matrix H_k in Eq. 8.5 may not be positive definite. In such a case the direction does not lead toward the minimum. Therefore, other methods have been developed so that the direction matrix is always positive definite. One such class of methods is the class of Quasi-Newton methods.

The quasi-Newton methods have a direction that is the solution of the following system of equations:

$$B_k d_k = -g_k \quad (8.7)$$

where $B_{k+1} = B_k + U_k$ where U_k is an updating matrix. B_0 often is taken to be the identity matrix, in which case the first step of the quasi-Newton method is a steepest descent step. Different methods of computing the update matrix lead to different algorithms, *e.g.*, Davidson-Fletcher-Powell (DFP) or Broyden-Fletcher-Goldfarb-Shannon (BFGS). Various modifications can be made to ensure that B_k is always positive definite. On each successive iteration, B_k acquires more information about the curvature of the function; thus, an approximate Hessian is computed, and users are spared the burden of programming an analytic Hessian. Both practical and theoretical considerations show that this approximate Hessian is generally quite good for the purpose of obtaining point estimates (Kelley, 1999, Section 4); whether this approximate Hessian can be used for computing standard errors of the point estimates is another matter entirely.

Let β^* represent the vector that minimizes the sum of squared residuals, and let H^* be the Hessian at that point. There do exist theorems which prove that $B_k \rightarrow H^*$ when the number of iterations is greater than the number of parameters. See, for example, Bazaara *et al.* (1993, p.322, Theorem 8.8.6). Perhaps on the basis of such theorems, it is sometimes suggested that the approximate Hessian provides reliable standard errors, *e.g.*, Bunday and Kiri (1987), Press, *et al.* (1992, p.419). However, the assumptions of such theorems are restrictive and quite difficult to verify in practice. Therefore, in practical situations it is not necessarily true that B_k resembles H_k (Gill, Murray and Wright, 1981, p.120). In fact, the approximate Hessian should *not* be used as the basis for computing standard errors. To demonstrate this important point, we consider a pair of examples from Wooldridge (2000). For both examples the package used is TSP v4.5, which employs automatic differentiation, and offers both the quasi-Newton method BFGS as well as a modified Newton-Raphson.

The first example (Wooldridge, p.538, Table 17.1) estimates a probit model with 'inlf' as the outcome variable (both examples use the dataset from Mroz (1987)). The PROBIT command with options HITER=F and HCOV=FNB uses the BFGS method to compute point estimates, and prints out standard errors using both the

Table 8.1 Probit Results for Mroz Data, Outcome Variable = 'inlf'

Variable	Coef.	Standard Errors Based On:					
		Approx. Hess.		Hessian		OPG	
C	0.270	0.511	(0.53)	0.509	(0.53)	0.513	(0.53)
NWIFEINC	-0.012	0.005	(-2.51)	0.005	(-2.48)	0.004	(-2.71)
EDUC	0.131	0.025	(5.20)	0.025	(5.18)	0.025	(5.26)
EXPER	0.123	0.019	(6.61)	0.019	(6.59)	0.019	(6.60)
EXPEFSQ	-0.002	0.001	(-3.15)	0.001	(-3.15)	0.001	(-3.13)
AGE	-0.053	0.008	(-6.26)	0.008	(-6.24)	0.009	(-6.12)
KIDSLT6	-0.868	0.118	(-7.35)	0.119	(-7.33)	0.121	(-7.15)
KIDSGE6	0.036	0.045	(0.81)	0.043	(0.83)	0.042	(0.86)

t-statistics in parentheses.

Table 8.2 Tobit Results for Mroz data, dep var = 'hours'

Variable	Coef.	Standard Errors Based On:					
		Approx. Hess.		Hessian		OPG	
C	965.3	0.415	(2327.1)	446.4	(2.16)	449.3	(2.14)
NWIFEINC	-8.814	0.004	(-2101.2)	4.459	(-1.98)	4.416	(-1.99)
EDUC	80.65	0.020	(4073.7)	21.58	(3.74)	21.68	(3.72)
EXPER	131.6	0.016	(8261.5)	17.28	(7.61)	16.28	(8.10)
EXPEFSQ	-1.864	0.001	(-3790.1)	0.538	(-3.47)	0.506	(-3.68)
AGE	-54.41	0.007	(-7978.7)	7.420	(-7.33)	7.810	(-6.97)
KIDSLT6	-894.0	0.105	(-8501.6)	111.9	(-7.99)	112.3	(-7.96)
KIDSGE6	-16.22	0.036	(-454.7)	38.64	(-0.42)	38.74	(-0.42)

t-statistics in parentheses.

approximation to the Hessian and the Hessian itself, as well as the OPG (outer product of the gradient) estimator from the BHHH (Berndt, Hall, Hall and Hausman, 1974) method for purposes of comparison. (The exact same point estimates are obtained when Newton's method is used.) The algorithm converged in 13 iterations. The results are presented in Table 8.1.

As can be seen by examining the *t*-statistics, all three methods of computing the standard error return similar results. The same cannot be said for the second example, in which Tobit estimation is effected for the same explanatory variables and 'hours' as the outcome variable (Wooldridge, p.544, Table 17.2). This time, the BFGS method converges in 15 iterations. Results are presented in Table 8.2.

Observe that the approximate Hessian standard errors are in substantial disagreement with the standard errors produced by the Hessian, the latter being about 1000

times larger than the former. The OPG standard errors, in contrast to the approximate Hessian standard errors, are in substantial agreement with the Hessian standard errors. It can be deduced that for the probit problem, the likelihood surface is very well-behaved, as all three methods of computing the standard error are in substantial agreement. By contrast, the likelihood surface for the tobit problem is not so well-behaved.

Thus far all the algorithms considered have been for unconstrained optimization. These algorithms can be applied to nonlinear least squares, but it is often better to apply specialized nonlinear least squares algorithms. The reason for this is that in the case of nonlinear least squares, the gradient and the Hessian have specific forms that can be used to create more effective algorithms. In particular, for NLS the gradient and Hessian are given by

$$g(\hat{\beta}) = J(\hat{\beta})'f(\hat{\beta}) \quad (8.8)$$

$$H(\hat{\beta}) = J(\hat{\beta})'J(\hat{\beta}) + Q(\hat{\beta}), \quad Q(\hat{\beta}) = \sum_{i=1}^n f_i(\hat{\beta})G_i(\hat{\beta}) \quad (8.9)$$

where $J(\hat{\beta})$ is the Jacobian matrix, $f(\hat{\beta})$ is the vector of residuals and G_i is the i -th contribution-to-the-Hessian matrix.

These specialized methods are based on the assumption that $Q(\hat{\beta})$ can be neglected, *i.e.*, that the problem is a *small-residual problem*. By small residual problem is meant that $\|f(\beta^*)\|$ is smaller than the largest eigenvalue of $J'(\beta^*)J(\beta^*)$. If, instead, they are of the same size, then there is no advantage to using a specialized method.

One specialized method is the Gauss-Newton method, which uses $J'J$ as an approximation to the Hessian; this is based on the assumption that $Q(\hat{\beta})$ is negligible. In combination with a line search, it is called *damped Gauss-Newton*. For small residual problems, this method can produce very rapid convergence; in the most favorable case, it can exhibit quadratic convergence, even though it uses only first derivatives. Its nonlinear maximum likelihood analogue is called the BHHH method. Another specialized method is the Levenberg-Marquardt method, which is an example of a *trust region algorithm*. The algorithms presented so far compute a direction and then choose a step length. A trust region method first computes a step length and then determines the direction of the step. The nonlinear maximum likelihood analogue is the quadratic hill-climbing method of Goldfeld, Quandt and Trotter (1966).

Finally, we note that mixed-methods can be very effective. For example, use a quasi-Newton method, with its wider radius of convergence, until the iterate is within the domain of attraction for the Newton method, and then switch to the Newton method.

8.3 SOME NUMERICAL DETAILS

It is important to distinguish between an algorithm and its implementation. The former is a theoretical approach to a problem and leaves many practical details unanswered. The latter is how the approach is applied practically. Two different implementations

of the same algorithm can produce markedly different results. For example, a damped quasi-Newton method only dictates that a line search be used; it does not specify how the line search is to be conducted. The Newton-Raphson method only dictates that second-derivatives are to be used; it does not specify how the derivatives are to be calculated: by forward differences, central differences, or analytically.

Consider the forward difference approximation to the derivative of a univariate function: $f'(x) = (f(x+h) - f(x))/h + R(h)$ where $R(h)$ is the remainder. This contrasts sharply with the central difference approximation: $f'(x) = (f(x+h) - f(x-h))/2h + R(h)$. For the function $f(x) = x^3$, it is easy to show that the remainder for the forward difference is $3x^2 + 3xh + h^2$ whereas for the central difference it is only $3x^2 + h^2$. Generally, the method of central differences produces smaller remainders, and thus more accurate derivatives, than the method of forward differences. Of course, with analytic derivatives the remainder is zero.

Another crucial point that can lead to different results is that the iterative process defined by Eq. 1 needs some basis for deciding when to stop the iterations. Some common termination criteria are:

- $|\text{SSR}^{k+1} - \text{SSR}^k| < \varepsilon_s$; when the successive change in the sum of squared residuals is less than some small value
- $\max_i [|\hat{\beta}_i^{k+1} - \hat{\beta}_i^k|] < \varepsilon_p$; when the largest successive change in some coefficient is less than some small value
- $\|g(\hat{\beta}^k)\| < \varepsilon_g$; when the magnitude of the gradient is less than some small value
- $g'H^{-1}g < \varepsilon$; this criterion involves both the gradient and the Hessian

The first three criteria are scale-dependent, that is, they depend upon the units of measurement. There do exist many other stopping rules, some of which are scale-independent versions of first three criteria above. The important point is that termination criteria must not be confused with convergence criteria.

Convergence criteria are used to decide whether a candidate point is a minimum. For example, at a minimum, the gradient must be zero, and the Hessian must be positive definite. A problem with many solvers is that they conflate convergence criteria and stopping rules, *i.e.*, they treat stopping rules as if they were convergence criteria. It is obvious, however, that while the stopping rules listed above are necessary conditions for a minimum, none or even all of them together constitutes a sufficient condition. Consider a minimum that occurs in a flat region of the parameter space: successive changes in the sum of squared residuals will be small at points that are far from the minimum. Similarly, parameters may not be changing much in such a region. In a flat region of the parameter space, the gradient may be very close to zero but, given the inherent inaccuracy of finite precision computation, there may be no practical difference between a gradient that is “close to zero” and one that is numerically equivalent to zero.

Finally, the user should be aware that different algorithms can differ markedly in the speed with which they approach a solution, especially in the final iterations.

Algorithms like the (modified) Newton-Raphson, that make full use of curvature information, converge very quickly. In the final iterations they exhibit *quadratic convergence*. At the other extreme, algorithms like steepest descent exhibit *linear convergence*. Between these two lie the quasi-Newton methods, which exhibit *super-linear convergence*. To make these concepts concrete, define $h^k = \beta^k - \beta^*$ where β^* is a local minimum. The following sequences can be constructed

$$\begin{array}{lll} \text{linear} & \|h^{k+1}\|/\|h^k\| \leq c & h^{k+1} = O(\|h^k\|) \\ \text{superlinear} & \|h^{k+1}\|/\|h^k\| \rightarrow 0 & h^{k+1} = o(\|h^k\|) \\ \text{quadratic} & \|h^{k+1}\|/\|h^k\|^2 \leq c & h^{k+1} = O(\|h^k\|^2) \end{array}$$

Nocedal and Wright (1999, p.199) give an example for the final few iterations of a steepest descent, BFGS, and Newton algorithm all applied to the same function. Their results are presented in Table 8.3. Observe that the Newton method exhibits quadratic convergence with the final few steps: e-02, e-04, e-08. Conversely, steepest descent is obviously converging linearly, while the quasi-Newton method BFGS falls somewhere in between. These rates of convergence apply not only to the parameters, but also to the value of the objective function, *i.e.*, the sum of squared residuals or the log likelihood. In the latter case, simply define $h^k = \text{LogL}^k - \text{LogL}^*$ where LogL^* is the value at the maximum.

Table 8.3 Convergence Rates

steepest descent	BFGS	Newton
1.827e-04	1.70e-03	3.48e-02
1.826e-04	1.17e-03	1.44e-02
1.824e-04	1.34e-04	1.82e-04
1.823e-04	1.01e-06	1.17e-08

Because quadratic convergence in the final iterations is commonly found in solutions obtained by the Newton method, if a user encounters only superlinear convergence in the final iterations of a Newton method, the user should be especially cautious about accepting the solution. Similarly, if a quasi-Newton method exhibits only linear convergence, the user should be skeptical of the solution.

8.4 WHAT CAN GO WRONG?

A good example is the Misra1a problem from the nonlinear suite of the StRD when it is given to the Microsoft Excel Solver. Not only is it lower difficulty problem, it is a two-parameter problem, which lends itself to graphical exposition. The equation is

$$y = \beta_1(1 - \exp(-\beta_2 x)) + \epsilon \quad (8.10)$$

with the fourteen observations given in Table 8.4.

Table 8.4 Data for Misra la Problem

Obs.	y	x	Obs.	y	x
1	10.070	77.60	8	44.820	378.40
2	14.730	114.90	9	50.760	434.80
3	17.940	141.10	10	55.050	477.30
4	23.930	190.80	11	61.010	536.80
5	29.610	239.90	12	66.400	593.10
6	35.180	289.00	13	75.470	689.10
7	40.020	332.80	14	81.780	760.00

The `Excel Solver` is used to minimize the sum of squared residuals, with `Start I` starting values of 500 for β_1 and 0.0001 for β_2 . The `Excel Solver` offers various options. The default method of derivative calculation is forward differences, with an option for central differences. The default algorithm is an unspecified “Newton” method, with an option for an unspecified “Conjugate” method. On `Excel 97` the default convergence tolerance (“ct”) is 0.001, though whether this refers to successive changes in the sum of squared residuals, coefficients, or some other criterion is unspecified. There is also an option for “automatic scaling” which presumably refers to recentring and rescaling the variables—this can sometimes have a meliorative effect on the ability of an algorithm to find a solution (Nocedal and Wright, 1999, p.27, and 94). Using `Excel 97`, five different sets of options were invoked:

- Options A: ct = 0.001 (`Excel 97` default)
- Options B: ct = 0.001 and automatic scaling
- Options C: ct = 0.0001 and central derivatives
- Options D: ct = 0.0001 and central derivatives and automatic scaling
- Options E: ct = 0.00001 and central derivatives and automatic scaling

For each set of options, the `Excel Solver` reported that it had found a solution. (`Excel 97`, `Excel 2000` and `Excel XP` all produced the same answers.) These five solutions are given in Table 8.5, together with the certified values from NIST. The correct digits are underlined. For example, Solutions A and B have no correct digits, while Solutions C and D each have the first significant digit correct.² Solution E has four correct digits for each of the coefficients, and five for the SSR. Additionally, for each of the six points the gradient of the objective function at that point is presented in brackets. These gradient values were produced three independent ways: via the nonlinear least squares command in `TSP`—taking care to note that `TSP` scales its

²The *significant digits* are those digits excluding leading zeroes.

Table 8.5 “Solutions” for the Misra1a Problem found by Excel Solver

	$\hat{\beta}_1$	$\hat{\beta}_2$	SSR
NIST	<u>238.94212918</u> [-1.5E-9]	<u>.00055015643181</u> [-0.00057]	<u>.12455138894</u>
Options A (8 iterations)	454.12442033 [0.23002]	.00026757574438 [420457.7]	16.725122137
Options B (8 iterations)	552.84275702 [0.16068]	.00021685528323 [454962.9]	23.150576131
Options C (31 iterations)	<u>244.64697774</u> [-0.00744]	.000 <u>5</u> 3527479056 [-0.00011]	<u>.16814681493</u>
Options D (33 iterations)	<u>241.96737442</u> [0.09291]	.000 <u>5</u> 4171455690 [37289.6]	<u>.15384239922</u>
Options E (37 iterations)	<u>238.93915212</u> [-5.8E-5]	<u>.00055016470282</u> [-23.205]	<u>.12455140816</u>

Accurate digits underlined, component of gradient in brackets.

gradient by $SSR/(n-2)$; by implementing equation (8) in the package R, which can produce $J(\hat{\beta})$ and $f(\hat{\beta})$; and programming from first principles using Mathematica. All three methods agreed to several digits. A contour plot showing the five solutions as well as the starting values (labeled “S”) is given in Figure 8.1.

It is also useful to consider the trace of each “solution”, as well as the trace of an accurate solution produced by the package S-Plus v6.2, which took 12 iterations to produce a solution with 9, 9 and 10 digits of accuracy for β_1 , β_2 and SSR, respectively. Of import is the fact that the S-Plus solver employs a Gauss-Newton algorithm, and this is a small-residual problem. The final five sum-of-squared residuals, as well as the difference of each from its final value, are given in Table 8.6.

Several interesting observations can be made about the Excel Solver solutions presented in Table 5. First, only Solution E might be considered a correct solution, and even the second component of its gradient is far too large.³ Solutions A and B have no accurate digits. Observe that the gradient of Solution C appears to be zero, but examining the sum of squared residuals shows that the gradient obviously is “not zero enough” (*i.e.*, 0.1681 is not nearly small enough). The gradient at solution E is not nearly zero, but it clearly has a smaller sum of squared residuals than solution C, and so despite its larger gradient may be preferred to solution C. This demonstrates

³The determination of “too large” is made with benefit of 20-20 hindsight. See §5.1

Fig. 8.1 Contours

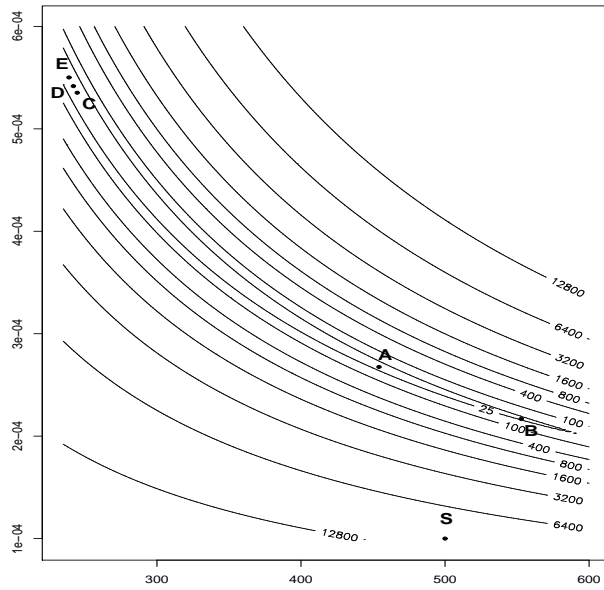


Table 8.6 Comparison of Solutions

	A	B	C	D	E	S-Plus
Residuals	18.649	29.133	0.17698	0.8774	0.1538	5770.4200
	18.640	28.966	0.17698	0.5327	0.1367	1242.3170
	18.589	27.670	0.16900	0.3972	0.1257	1.1378420
	18.310	27.026	0.16900	0.2171	0.1246	0.1245559
	16.725	23.151	0.16815	0.1538	0.1245	0.1245514
Differences	1.924	5.982	0.00883	0.7236	0.0293	5770.2954
	1.915	5.815	0.00883	0.3789	0.0122	1242.1924
	1.864	4.519	0.00085	0.2434	0.0012	1.0132906
	1.585	3.875	0.00085	0.0633	0.0001	0.0000045
	0.000	0.000	0.00000	0.0000	0.0000	0.0000000

the folly of merely examining gradients (and Hessians); examination of the trace also can be crucial.

The `Excel Solver` employs an unspecified Newton algorithm, with an unknown convergence rate. Rather than assume that this method is quadratically convergent, let us assume that it is superlinearly convergent. Examining Table 6, all the `Excel` solutions exhibit linear convergence, even Solution E, for which $h_{k+1} \approx 0.1h_k$. In particular, examining the trace of Solution C shows that the Solver is searching in a very flat region that can be characterized by at least two plateaus. Even though each component of the gradient appears to be zero, the trace does not exhibit the necessary convergence, so we do not believe Point C to be a solution.

Panel 1 of Figure 8.2 does not show sufficient detail, and some readers may think, especially given the gradient information provided in Table 8.6, that point C is a local minimum. It is not, as shown clearly in Panel 2.

We have just analyzed five different solutions from one package. It is also possible to obtain five different solutions from five different packages, something Stokes (2003) accomplished when trying to replicate published probit results. It turned out that for the particular data used by Stokes, the maximum likelihood estimator did not exist. This did not stop several packages from reporting that their algorithms had converged to a solution—a solution that did not exist! His paper is instructive reading.

8.5 FOUR STEPS

Stokes was not misled by his first solver because he did not accept uncritically its output. In fact, critically examining its output is what led him to use a second package. When this produced a different answer, he knew something was definitely wrong. And this was confirmed by the different answers from his third, fourth and fifth packages. With his first package, Stokes varied the algorithm, the convergence tolerance, starting values, etc. This is something that every user should do with every nonlinear estimation. Suppose that a user has done this and identified a possible solution. How might he verify the solution? McCullough and Vinod (2003) recommend four steps:

1. examine the gradient—is $\|g\| \approx 0$?
2. inspect the sequence of function values—does it exhibit the expected rate of convergence?
3. analyze the Hessian—is it positive semidefinite?—is it ill-conditioned?
4. profile the objective function—is the function approximately quadratic?

Gill, Murray and Wright (1981, p.313) note that if the first three conditions hold, then very probably a solution has been found, regardless of whether the program has declared convergence. The fourth step justifies the use of the usual t -statistics for coefficients reported by most packages. These t -statistics are Wald-statistics and, as such, are predicated on the assumption that the objective function, in this case the sum-of-squares function, is approximately quadratic in the vicinity of the minimum. If the

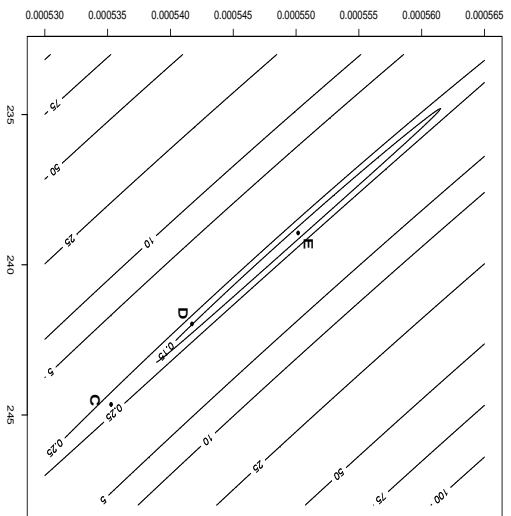
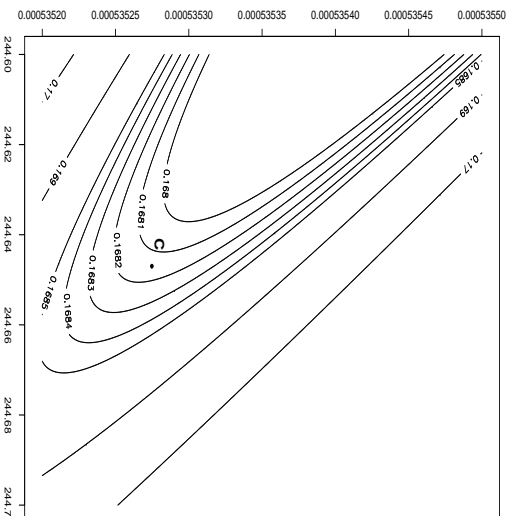


Fig. 8.2 Contours



function is not approximately quadratic, then the Wald statistic is invalid and other methods are more appropriate, *e.g.*, likelihood ratio intervals; this topic is addressed in detail in Section Six. The easy way to determine whether the objective function is approximately quadratic is to profile the objective function. Each of these four steps is discussed in turn.

8.5.1 Step One: examine the gradient

At the solution, each component of the gradient should be zero. This is usually measured by the magnitude of the gradient, squaring each component, summing them, and taking the square root of the sum. The gradient will only be zero to the order of the covariate(s). To see this, for the Misrala problem, multiply the y and x vectors by 100 and 10, respectively. The correct solution changes from $\beta_1^* = 238.94212918$, $\beta_2^* = 0.00055015643181$, $SSR = 0.12455138894$ to $\beta_1^* = 23894.212918$, $\beta_2^* = 0.000055015643181$, $SSR = 1245.5138894$ and the gradient changes from $[-1.5E-9, -0.000574]$ to $[1.5E-7, -57.4]$. The moral is that what constitutes a zero gradient depends on the scaling of the problem. See Gill, Murray and Wright (1981, Section 7.5 and Section 8.7) for discussions of scaling.

Of course, a package that does not permit the user to access the gradient is of little use here.

8.5.2 Step Two: inspect the trace

Various algorithms have different rates of convergence. By rate of convergence is meant the rapidity with which the function value approaches the extremum as the parameter estimates get close to the extremal estimates. As an example, let β^* be the vector that minimizes the sum of squares for a particular problem. If the algorithm in question is Newton-Raphson, which has quadratic convergence, then as $\hat{\beta}^k \rightarrow \beta^*$ the sum of squared residuals will exhibit quadratic convergence as shown in Table 8.3.

Suppose, then, that Newton-Raphson is used, and the program declares convergence. However, the trace exhibits only linear convergence in its last few iterations. Then it is doubtful that a true minimum has been found. This type of behavior can occur when, for example, the solver employs “parameter convergence” as a termination criterion, and the current parameter estimate is in a very flat region of the parameter space. Then it makes sense that the estimated parameters are changing very little, and neither is the function value when the solver ceases iterating.

Of course, a solver that does not permit the user to access the function value is of little use here.

8.5.3 Step Three: analyze the Hessian

The requirement for a multivariate minimum, as in the case of minimizing a sum-of-squares function, is that the gradient is zero and the Hessian is positive definite. The easiest way to check the Hessian is to do an eigensystem analysis and make sure

all the eigenvalues are positive. The user should be alert to the possibility that his package does not have accurate eigen routines. If the developer of the package does not offer some positive demonstration that the package's matrix routines are accurate, then the user should request proof.

In the case of a symmetric definite matrix, *e.g.* the covariance matrix, the ratio of the largest eigenvalue to the smallest eigenvalue is the *condition number*. If this number is high, then the matrix is said to be "ill conditioned." The consequences of this ill-conditioning are three-fold. First, it indicates that the putative solution is in a "flat" region of the parameter space, so that some parameter values can change by large amounts while the objective function changes hardly at all. This situation can make it difficult to locate and to verify the minimum of a NLS problem (or the maximum of a NML problem). Second, this ill-conditioning leads to serious cumulated rounding error and a loss of accuracy in computed numbers. As a general rule, when solving a linear system, one digit of accuracy is lost for every power of ten in the condition number (Judd, 1998, p.68). A PC has 16 digits. If the condition number of the Hessian is on the order of 10^{**9} , then the coefficients will be accurate to no more than seven digits. Third, McCullough and Vinod (2002) show that if the Hessian is ill-conditioned, then the quadratic approximation fails to hold in at least one direction. Thus, a finding in Step Three that the Hessian is ill-conditioned automatically implies that Wald inference will be unreliable for at least one of the coefficients.

Of course, a package that does not permit the user to access the Hessian (perhaps because it cannot even compute the Hessian) is of little use here.

8.5.4 Step Four: profile the objective function

The first three steps were concerned with obtaining reliable point estimates. Point estimates without some measure of variability are meaningless, and so reliable standard errors also are of interest. The usual standard errors produced by nonlinear routines, the so-called *t*-statistics, are more formally known as Wald statistics. For their validity, they depend on the objective function being quadratic in the vicinity of the solution (Box and Jenkins, 1976). Therefore, it is also of interest to determine whether, in fact, the objective function actually is locally quadratic at the solution. To do this, profile methods are very useful.

The essence of profiling is simplicity itself. Consider a nonlinear problem with three parameters, α , β and θ , and let $\hat{\alpha}^*$, $\hat{\beta}^*$ and $\hat{\theta}^*$ be the computed solution. The objective function (LogL for MLE, SSR for NLS) has value f^* at the solution. For sake of exposition, suppose $\hat{\alpha}^* = 3$ with a standard error of $se(\alpha) = 0.5$ and a profile of α is desired. For plus/minus some number of standard deviations, say four, choose several values of α , *e.g.*, $\alpha_1 = 1.0, \alpha_2 = 1.5, \alpha_3 = 2.0, \dots, \alpha_7 = 4.0, \alpha_8 = 4.5, \alpha_9 = 5.0$. Fix $\alpha = \alpha_1$, re-estimate the model allowing β and θ to vary, and obtain the value of the objective function at this new (constrained) extremum, f_1 . Now fix $\alpha = \alpha_2$ and obtain f_2 . The sequence of pairs $\{\alpha_i, f_i\}$ traces out the profile of $\hat{\alpha}$. If the profile is quadratic, then Wald inference for that parameter is justified.

Table 8.7 Results of Profiling β_1

i	β_1	$\delta(\beta_1)$	SSR	$\tau(\beta_1)$
9	244.3561	2.0	0.1639193	1.947545
8	243.0026	1.5	0.1469733	1.469780
7	241.6491	1.0	0.1346423	0.986010
6	240.2956	0.5	0.1271061	0.496118
5	238.9421	0.0	0.1245514	0.000000
4	237.5886	-0.5	0.1271722	-0.502497
3	236.2351	-1.0	0.1351700	-1.011463
2	234.8816	-1.5	0.1487542	-1.527035
1	233.5280	-2.0	0.1681440	-2.049382

Visually it is easier to discern deviations from linear shape than deviations from quadratic shape. The following transformation makes it easier to assess the validity of the quadratic approximation.

$$\tau(\alpha) = \text{sign}(\alpha - \hat{\alpha}^*)\sqrt{S(\alpha) - S(\hat{\alpha}^*)}/s \quad (8.11)$$

where s is the standard error of the estimate.⁴

A plot of $\tau(\alpha_i)$ against α_i will be a straight line if the quadratic approximation is valid. Now, let the studentized parameter be

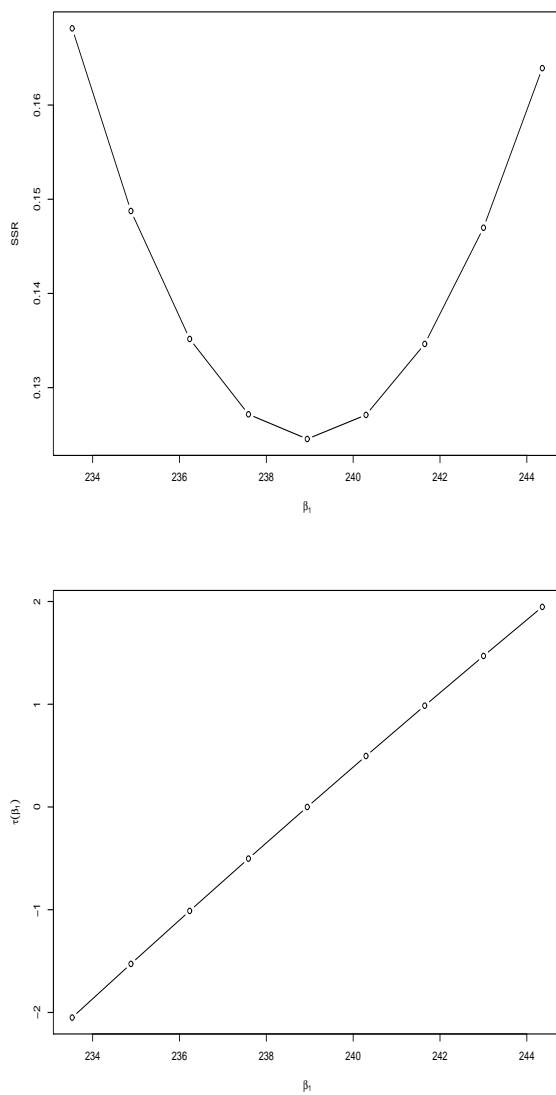
$$\delta(\alpha_i) = \frac{\alpha_i - \hat{\alpha}^*}{\text{se}(\alpha_i)} \quad (8.12)$$

A plot of $\tau(\alpha_i)$ against $\delta(\alpha_i)$ will be a straight line with unit slope through the origin.

As a concrete example, consider profiling the parameter β_1 from the StRD Misra1a problem, the results of which are given in Table 8.7. Figure 8.3 shows the plot of SSR vs. β_1 to be approximately quadratic, and the plot of $\tau(\beta_1)$ vs. β_1 is a straight line. Many packages plot τ vs. the parameter instead of τ vs. δ . Usually it is not worth the trouble to convert the former to the latter though, on occasion, it may be necessary to achieve insight into the problem.

Profile methods are discussed at length in Bates and Watts (1988, Section 6) and Venables and Ripley (1999, Section 8.5). Many statistical packages offer them, e.g., SAS and S-PLUS. Many econometrics packages don't, although GAUSS is an exception. What a user should do if he finds that Wald inference is unreliable is the subject of the next section.

⁴For NML, replace the right hand side of Equation 8.11 with $\text{sign}(\alpha - \hat{\alpha}^*)\sqrt{2(\text{Log}L^* - \text{Log}L)}$.

Fig. 8.3 Profiles of β_1 vs. SSR and β_1 vs. $\tau(\beta_1)$ 

8.6 WALD VS. LIKELIHOOD INFERENCE

It is commonly thought that Wald inference and likelihood ratio (LR) inference are equivalent, *e.g.*, Rao (1973, p.418). As noted earlier, the Wald interval is valid only if the objective function is quadratic. However, the LR interval is much more generally valid. In the scalar case, the 95% Wald interval is given by

$$\hat{\theta} \pm 1.96\text{se}(\hat{\theta}) \quad (8.13)$$

while the LR interval is given by

$$\{\theta, 2 \ln \frac{L(\hat{\theta})}{L(\theta)} \leq 3.84\} \quad (8.14)$$

The Wald interval is exact if

$$\frac{\hat{\theta} - \theta}{\text{se}(\hat{\theta})} \sim N(0, 1) \quad (8.15)$$

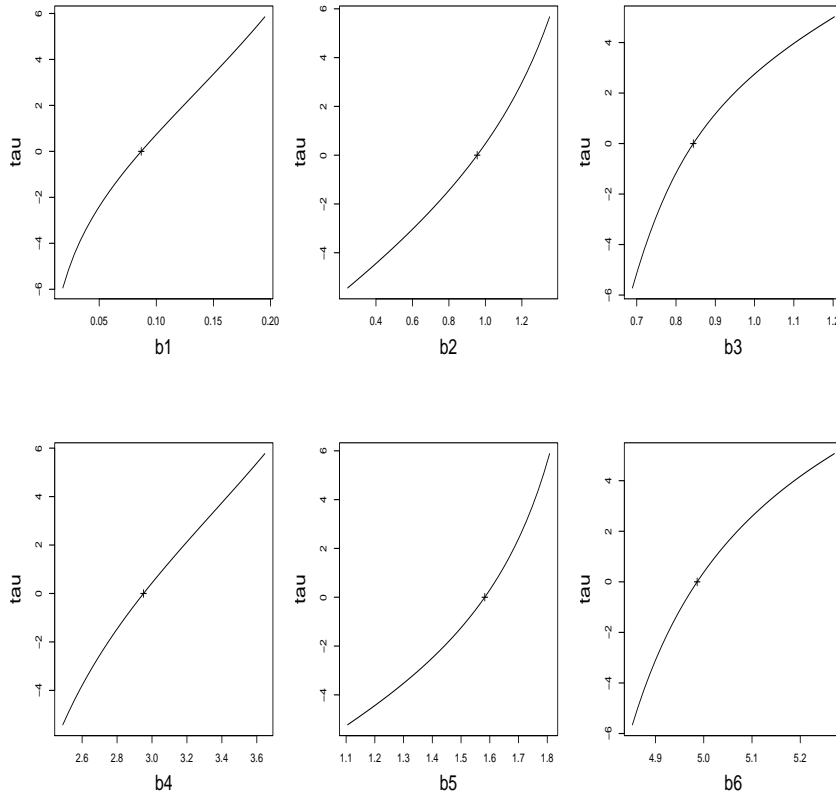
while the LR interval is exact so long as there exists some transformation $g(\cdot)$ such that

$$\frac{g(\hat{\theta}) - g(\theta)}{\text{se}(g(\hat{\theta}))} \sim N(0, 1) \quad (8.16)$$

and it is not necessary that the specific function $g(\cdot)$ be known, just that it exists. Consequently, when the Wald interval is valid, so is the LR interval, but not conversely. Thus, Gallant's (1987, p.147) advice is to "avoid the whole issue as regards inference and simply use the likelihood ratio statistic in preference to the Wald statistic."

The assertion that the LR intervals are preferable to the Wald intervals merits justification. First, a problem for which the profiles are nonlinear is needed. Such a problem is the six parameter Lanczos3 problem from the StRD, for which the profiles were produced using the package `R(Ihaka and Gentleman, 1996)`, with the "profile" command from the `MASS` library of Venables and Ripley (1999). These profiles are presented in Figure 8.4. None of the profiles is remotely linear, so it is reasonable to expect that LR intervals will provide better coverage than Wald intervals. To assess this claim, a Monte Carlo study is in order.

Using the NIST solution as true values and a random generator with mean zero and standard deviation equal to the standard error of the estimate of the NIST solution, 3999 experiments are run. For each run, 95% intervals of both Wald and LR type are constructed. The LR intervals are computed using the 'confint.nls' command, which actually only approximates the likelihood ratio interval. The proportion of times that an interval contains the true parameter is its *coverage*. This setup is repeated for each of three types of errors: normal, Student's-*t* with 10 df (rescaled), and a chi-square with 3 df (recentered and rescaled). The results are presented in Table 8.8. As can be seen, for each type of error and for each parameter, the LR interval provides substantially better coverage than the Wald interval.

Fig. 8.4 Profiles for Lanczos3 Parameters

8.7 CONCLUSIONS

Software packages frequently have *default options* for their nonlinear solvers. For example, the package may offer several algorithms for nonlinear least squares, but unless the user directs otherwise, the package will use Gauss-Newton. As another example, the convergence tolerance might be $1E-3$; perhaps switching the tolerance to $1E-6$ would improve the solution. It should be obvious that “solutions” produced by use of default options should not be accepted by the user until the solution has been verified by the user; see McCullough (1999).

Though many would pretend otherwise, nonlinear estimation is far from automated, even with today’s sophisticated software. There is more to obtaining trustworthy estimates than simply tricking a software package into declaring convergence. In fact, when the software package declares convergence, the researcher’s job is just beginning – he has to verify the solution offered by the software. Software packages differ markedly not only in their accuracy, but also in their ability to verify potential

Table 8.8 Monte Carlo Results

Errors	Test	b1	b2	b3	b4	b5	b6
Normal	Wald	91.725	92.125	92.650	92.300	92.550	92.875
	LR	94.200	94.125	94.275	94.175	94.200	94.275
$t(10)$	Wald	92.550	93.200	93.250	92.875	93.200	93.275
	LR	95.025	94.975	94.800	94.975	94.825	94.825
$\chi^2(3)$	Wald	92.700	93.300	93.275	93.500	93.075	93.450
	LR	95.275	95.250	95.550	95.425	95.625	95.475

Coverage for 95% Confidence Intervals

solutions. A desirable software package is one that makes it easy to guard against false convergence. Some relevant features are as follows:

- The user should be able to specify starting values.
- For nonlinear least squares, at least two algorithms should be offered: a modified Newton and a Gauss-Newton; a Levenberg-Marquardt makes a good third. The NL2SOL algorithm (Dennis, Gay and Welsch, 1981) is highly regarded. For unconstrained optimization (*i.e.*, for nonlinear maximum likelihood), at least two algorithms should be offered: a modified Newton and the BFGS. Again, the Bunch, Gay and Welsch (1993) algorithm is highly regarded.
- For nonlinear routines, the user should be able to fix one parameter and optimize over the rest of the parameters, in order to calculate a profile (all the better if the program has a “profile” command).
- The package should either offer LR statistics, or enable the user to write such a routine.
- For routines that use numerical derivatives, the user should be able to supply analytic derivatives. Automatic differentiation is very nice to have when dealing with complicated functions.
- The user should be able to print out the gradient, the Hessian, and the function value at every iteration.

Casually perusing scholarly journals, and briefly scanning those articles that conduct nonlinear estimation, will convince the reader of two things. First, many researchers run their solvers with the default settings. This, of course, is a recipe for disaster, as was discovered by team of statisticians working on a large pollution study (Revkin, 2002). They simply accepted the solution from their solver, making no attempt whatsoever to verify it, and wound up publishing an incorrect solution. Second, even researchers who do not rely on default options practically never attempt to verify

the solution. One can only wonder how many incorrect nonlinear results have been published.

REFERENCES

- Bates, Douglas M. and Donald G. Watts. (1988). *Nonlinear Regression Analysis and Its Applications*. New York: Wiley
- Bazaraa, M., H. Sherali and C. Shetty. (1993). *Nonlinear Programming: Theory and Algorithms*. Second Edition. New York: Wiley
- Berndt, E., R. Hall, B. Hall and J. Hausman. (1974). Estimation and Inference in Nonlinear Structural Models. *Annals of Economic and Social Measurement* (3/4), 653-665
- Box, G. E. P. and G. Jenkins. (1976). *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day
- Bunch, D. S., D. M. Gay, and R. E. Welsch. (1993). Algorithm 717: Subroutines for Maximum Likelihood and Quasi-Likelihood Estimation of Parameters in Nonlinear Regression Models. *ACM Trans. Math. Software* **19**, 109-130
- Bunday, Brian D. and Victor A. Kiri. (1987). Maximum Likelihood Estimation—Practical Merits of Variable Metric Optimization Methods. *The Statistician* **36**, 349-355.
- Dennis, Jr., J. E. (1984). A User's Guide to Nonlinear Optimization Algorithms. *Proceedings of the IEEE* **72**, 1765-1776.
- Dennis, Jr., J. E., D. M. Gay, and R. E. Welsch. (1981). ALGORITHM 573: NL2SOL—An Adaptive Nonlinear Least-Squares Algorithm. *ACM Trans. Math. Software* **7**, 369-383.
- Dennis, Jr., J. E. and Robert B. Schnabel. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia: SIAM.
- Fletcher, R. (1987). *Practical Methods of Optimization*. Second Edition. New York: Wiley & Sons.
- Gallant, A. Ronald. (1987). *Nonlinear Statistical Models*. New York: Wiley.
- Gill, Phillip E., Walter Murray, M. H. Wright. (1981). *Practical Optimization*. London: Academic Press.
- Goldfeld, S., R. Quandt, and H. Trotter. (1966). Maximisation by Quadratic Hill-Climbing. *Econometrica* **34**, 541-551.

- Hendry, David F. (1995). *Dynamic Econometrics*. New York: Oxford University Press.
- Ihaka, R. and R. Gentleman. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics* **5**, 299-314.
- Judd, K. (1998). *Numerical Methods in Economics*. Cambridge, MA: MIT Press.
- Kelley, C. T. (1999). *Iterative Methods for Optimization*. Philadelphia: SIAM.
- Kennedy, W. J. and Gentle, J. E. (1980). *Statistical Computing* New York: Marcel Dekker.
- McCullough, B. D. (1998). Assessing the Reliability of Statistical Software: Part I. *The American Statistician* **52**, 358-366.
- McCullough, B. D. (1999). Assessing the Reliability of Statistical Software: Part II. *The American Statistician* **53**, 149-159.
- McCullough, B. D. and Charles G. Renfro. (1999). Benchmarks and Software Standards: A Case Study of GARCH Procedures. *Journal of Economic and Social Measurement* **25**, 59-71.
- McCullough, B. D. and Charles G. Renfro. (2000). Some Numerical Aspects of Nonlinear Estimation. *Journal of Economic and Social Measurement* **26**, 63-77.
- McCullough, B. D. and H. D. Vinod. (1999). The Numerical Reliability of Econometric Software. *Journal of Economic Literature* **37**, 633-665
- McCullough, B. D. and H. D. Vinod. (2003). Verifying the Solution from a Nonlinear Solver: A Case Study. *American Economic Review*, (to appear).
- McCullough, B. D. and Berry Wilson. (1999). On the Accuracy of Statistical Procedures in Microsoft Excel 97. *Computational Statistics and Data Analysis* **31**, 27-37
- McCullough, B. D. and Berry Wilson. (2002). On the Accuracy of Statistical Procedures in Microsoft Excel 2000 and Excel XP. *Computational Statistics and Data Analysis*, (to appear.)
- Monahan, John F. (2001). *Numerical Methods of Statistics*. New York: Cambridge University Press.
- Mroz, Thomas A. (1987). The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions. *Econometrica* **55**, 765-799
- Nocedal, Jorge and Stephen J. Wright. (1999). *Numerical Optimization*. New York: Springer-Verlag.

- Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. (1986). *Numerical Recipes in FORTRAN*. Cambridge: Cambridge University Press.
- Rao, C. R. (1973). *Linear Statistical Inference and Its Applications*. New York: Wiley & Sons.
- Revkin, Andrew (2002). "Data Revised on Soot in Air and Deaths," *New York Times* (National Edition) 5 June 2002, p. A23
- Seber, G. A. F. and C. J. Wild. (1999). *Nonlinear Regression*, New York: Wiley
- Stokes, Houston H. (2003). On the Advantage of Using Two or More Econometric Software Systems to Solve the Same Problem. *Journal of Economic and Social Measurement*. (to appear)
- Venables, W. N. and B. D. Ripley. (1999). *Modern Applied Statistics Using S-Plus*. Third Edition. New York: Springer
- Wooldridge, Jeffrey M. (2000). *Introductory Econometrics: A Modern Approach*. Mason, OH: South-Western Publishing.