NBER WORKING PAPER SERIES

ROBUST NONLINEAR REGRESSION USING THE DOGLEG ALGORITHM

Roy E. Welsch*
Richard A. Becker**

Working Paper No. 76

COMPUTER RESEARCH CENTER FOR ECONOMICS AND MANAGEMENT SCIENCE
National Bureau of Economic Research, Inc.
575 Technology Square
Cambridge, Massachusetts 02139

March 1975

Preliminary: not for quotation

NBER working papers are distributed informally and in limited numbers for comments only. They should not be quoted without written permission.

This report has not undergone the review accorded official NBER publications; in particular, it has not yet been submitted for approval by the Board of Directors.

*NBER Computer Research Center and Massachusetts Institute of Technology, Sloan School of Management. Research supported in part by National Science Foundation Grant GJ-1154X3 to the National Bureau of Economic Research, Inc.

**Bell Telephone Laboratories. Research supported in part by National Science Foundation Grant GJ-1154X3 to the National Bureau of Economic Research, Inc.

Abstract

What are the statistical and computational problems associated with robust nonlinear regression? This paper presents a number of possible approaches to these problems and develops a particular algorithm based on the work of Powell and Dennis.

Contents

1. Introduction 1
2. The Problem 1
3. What Can the Average Man Do? 1
4. Nonlinear Reweighted Least-Squares 2
5. Creating Specialized Algorithms 3
6. Robust Nonlinear Regression 4
7. Starting Values 5
8. Scale Computation 5
9. Confidence Regions 5
10. Eliminating Second Derivatives 6
11. Examples 6
12. Concluding Remarks 6
13. References 6

Tables

Table 1. Marketing Model Data 7
Table 2. Marketing Model Results 8
Table 3. Test Function Results 8

ROBUST NONLINEAR REGRESSION USING THE DOGLEG ALGORITHM

1. INTRODUCTION

In recent years the concepts of robust estimation have lead to a rethinking of the ways we fit models to data. Papers by Beaton and Tukey [1974] and Andrews [1974] have proposed algorithms for robust linear regression using iteratively reweighted least-squares. This technique has proved to be quite successful and has considerable intuitive appeal because of its connection to weighted least-squares regression.

In late 1973 the authors designed and implemented a robust linear regression macro on the TROLL system at the NBER Computer Research Center. It makes use of reweighted least-squares, iterative scaling, optional starts including least absolute residuals, and provides a robust trace of the coefficients as a "robustness parameter" is varied. After some economists and management scientists had worked with this macro, we received a number of requests to provide similar facilities for nonlinear problems. In what follows we discuss several possible approaches to robust nonlinear regression, outline a few successful algorithms, and discuss our experience with them.

2. THE PROBLEM

Assume that we are interested in fitting the model $f(\theta)=\{f_1(\theta),\ldots,f_n(\theta)\}^T, \theta=\{\theta_1,\ldots,\theta_p\}$ to the data $y=\{y_1,\ldots,y_n\}^T$. We shall seek to do this by moving from some start $\theta^{(0)}$ toward a local minimum (with respect to θ) of

$$F_s(\theta) = \sum_{i=1}^{n} \rho_c\left(\frac{y_i-f_i(\theta)}{s}\right) \qquad (2.1)$$

where $\rho(\cdot)$ is assumed to satisfy $\rho(t) \le \rho(u)$ if $|t|<|u|$ and is often viewed as a loss function (which, in general, need not be independent of i or symmetric).

For nonlinear least-squares ($\rho(t)=t^2$) we have always faced the problem of specifying starting values. For robust loss functions such as

$$\rho_c(t) = \begin{cases} \frac{t^2}{2} & |t|\le c \\ c|t| - \frac{c^2}{2} & |t|> c \end{cases} \qquad (2.2)$$

$$\rho_c(t) = \begin{cases} c^2[1-\cos(t/c)] & |t|\le c\pi \\ 2c^2 & |t|> c\pi \end{cases} \qquad (2.3)$$

we not only need a starting value but, because these loss functions are not scale invariant, we also need a way to measure the scale (size is perhaps a better word in this context) of the residuals, $r(\theta)=y-f(\theta)$, at the beginning of the computation and, in some cases, to remeasure it as the computation continues. We must also choose c, the robustness parameter, or at least provide ways to indicate the effects of changing c.

We note that neither (2.2) nor (2.3) have a second derivative everywhere. There are approximations to these functions which do (e.g. the bisquare of Beaton and Tukey [1974]) but having a second derivative everywhere has not proved to be practically important for the algorithms we shall discuss.

3. WHAT CAN THE AVERAGE MAN DO?

We have found that many people have access to some form of general nonlinear optimization program and/or a special routine for nonlinear least-squares. Most of the researchers interested in robust fitting are not interested in extensively modifying these programs or writing new ones. So we discuss first some approaches to robust nonlinear regression

that allow the use of existing programs.

If we assume that a general nonlinear optimization routine is available then it seems reasonable to try to estimate the scale, s, by making it a part of the optimization problem,

$$\min_{\theta,s} \sum_{i=1}^{n} \rho_c\left(\frac{r_i(\theta)}{s}\right) + S_1 \log s \qquad (3.1)$$

in direct analogy to the related maximum likelihood problem. It is also immediately clear that this idea will fail for robust loss functions which are bounded (such as (2.3)) because s will be forced to 0. (There is no proper maximum likelihood model in this case.) However for loss functions of the form (2.2) this is not true and (3.1) is viable.

The constant S_1 can be chosen in a variety of ways, one of which is the following. Differentiating (3.1) with respect to s and setting it equal to 0 we obtain

$$S_1 = \sum_{i=1}^{n} \rho_c'\left(\frac{r_i(\theta)}{s}\right)\frac{r_i(\theta)}{s} \quad . \qquad (3.2)$$

If the residuals were Gaussian then we might try to choose S_1 so that s would be asymptoticially unbiased giving

$$S_1 = (n-p)\int_{-\infty}^{\infty} t\rho_c'(t)\, d\Phi(t)$$

where $\Phi(t)$ denotes the standardized Gaussian distribution function.

Huber and Dutter [1974] have suggested a related idea. They propose replacing (3.1) by

$$\min_{\theta,s} \sum_{i=1}^{n} s\rho_c\left(\frac{r_i(\theta)}{s}\right) + S_2 s \qquad (3.3)$$

where

$$S_2 = (n-p)\int_{-\infty}^{\infty} t\rho_c'(t) - \rho_c(t)\, d\Phi(t) \quad .$$

When $\rho_c(t)$ is the Huber type, (2.2), then

$$t\rho_c'(t) - \rho_c(t) = (\rho_c'(t))^2/2$$

and the normal equation for s reduces to the scale estimate proposed by Huber [1964, 1973]. This idea also fails for bounded loss functions.

We have tried (3.1) and (3.3) using a general optimization algorithm to be described later and found that both work about equally well. Both can be implemented very quickly.

What about the bounded loss function case? It is natural to consider a penalty function to keep s positive. For example

$$\min_{\theta,s} \sum_{i=1}^{n} \rho_c\left(\frac{r_i(\theta)}{s}\right) + B_1 \log s + B_2/s. \qquad (3.4)$$

In effect, we have added the negative log likelihood of an inverted gamma prior distribution for the scale parameter.

We must, of course, specify B_1 and B_2. If there is prior information about the scale then B_1 and B_2 would be taken from that. Otherwise we would choose $B_1=S_1$ and B_2 by penalty function considerations such as those discussed by Bard [1974, p.145]. Our experience with this method is limited and not wholly satisfactory.

The above three methods provide ways for a person with a general nonlinear optimizer to simply put in an objective function different from the one for least-squares and use his program as is. We do not advise doing this blindly but it generally works (especially the first two methods). It has drawbacks. It is expensive because another parameter (s) must be estimated (with algorithms of order p^2) and the objective function is more complicated and numerically less pleasing. When choice is available, it does not seem reasonable to pay so much for the privilege of iteratively modifying s.

There is another obvious approach. That is simply to find an $s^{(0)}$ (see section 7) and then minimize (2.1) with $s=s^{(0)}$. At the end of this computation a new s, say $s^{(1)}$, is computed based on the current values of θ, then left fixed until a new local minimum is found, etc. until $s^{(k+1)}$ changes less than, say, ten percent from $s^{(k)}$. This is simple, but often proved to be as expensive as the earlier approaches. It does, however, lead us to consider ways to handle iterative scaling without making use of the objective function.

4. NONLINEAR REWEIGHTED LEAST-SQUARES

For those with special nonlinear least-squares algorithms available it is natural to attempt to adapt the iteratively reweighted least-squares idea mentioned earlier to the nonlinear problem. We now discuss this method in more detail.

The gradient and Hessian of $F_s(\theta)$ are

$$g_s(\theta) = -\frac{J^T(\theta)}{s}\rho'\left(\frac{r(\theta)}{s}\right) \qquad (4.1)$$

$$H_s(\theta) = \sum_{i=1}^{n} -\rho'\left(\frac{r_i(\theta)}{s}\right)\frac{\nabla^2 f_i(\theta)}{s} + \frac{J^T(\theta)}{s}\rho''\left(\frac{r(\theta)}{s}\right)\frac{J(\theta)}{s} \qquad (4.2)$$

where $J(\theta)=[\partial f_i(\theta)/\partial\theta_j]$ is the Jacobian matrix, $\nabla^2 f_i(\theta)=[\partial^2 f_i/\partial\theta_k\partial\theta_j]$, $\rho'(r(\theta)/s)=[\rho'(r_1(\theta)/s),\ldots,\rho'(r_n(\theta)/s)]^T$, and $\rho''(r(\theta)/s)$ is an nxn diagonal matrix. Now define a weight function $w(t)=\rho'(t)/t$ and an approximate Hessian

$$\tilde{H}_s(\theta) = \frac{1}{s^2}\left[\sum_{i=1}^{n} - w_i r_i \nabla^2 f_i + J^T w J\right]$$

where $w_i = w(\frac{r_i}{s})$, w is an nxn diagonal matrix, and $\rho''(t)$ is approximated by w(t). If we have starting values $\theta^{(0)}$ and $s=s^{(0)}$ then the first step of reweighted least-squares is

$$\theta^{(1)} = \theta^{(0)} + \tilde{H}_s^{-1}(\theta^{(0)})\, g_s(\theta^{(0)})$$

which in the linear case is

$$= \theta^{(0)} + (X^T w X)^{-1} X^T w\, (Y - X\theta^{(0)}) \ .$$

The whole procedure can, of course, be iterated. Except for the fact that $\rho''(t)$ has been approximated by w(t) this is just the first Newton step for the solution of (2.1) in the linear case. Using w(t) makes $\tilde{H}$ positive semi-definite and makes the analogy to weighted least-squares obvious.

A word of caution is in order. Even if X^TX is well conditioned, X^TwX may be very ill-conditioned and the first Newton step a very poor one. (This can happen when there are low weights on observations which contain all of the information about a parameter or information about how to separate the effects of two carriers.) Even if X^TwX is well behaved at a local minimum a bad start can lead to poor results. Often this problem is ignored in the linear case because of the availability of robust, scale invariant procedures, such as least absolute residuals [Barrodale and Roberts (1973)] to provide starting values. All of the literature about Newton-Raphson methods applies to this problem - such methods are only reasonable in a neighborhood of a local minimum. Good algorithms for robust regression should contain some diagnostic studies of the data matrix to determine potential high leverage observations. Varying the robustness parameter c can also be very useful. Ridge regression techniques could also be employed.

In the nonlinear case we do not have such good techniques for finding starting values and the first term of the Hessian does not vanish. But most nonlinear least-squares routines ignore the first term of the Hessian and use a technique like that of Marquardt (1963) to overcome the difficulties of Gauss-Newton steps away from the local minimum. Once in a region where the residuals are, hopefully, small the first term of the Hessian can be more safely ignored. Some work has been done on the large residual least-squares problem, e.g. Dennis [1973]. Robust loss functions help to reduce the size of the first term of the Hessian because $|\rho'(t)| < |t|$ for large residuals and with (2.3), $\rho'(t)$ is eventually zero.

With the same caveats we have always had in using Gauss-Marquardt nonlinear least-squares routines for very nonlinear problems, it is reasonable to propose that (2.1) be attacked by finding starting values $\theta^{(0)}$ and $s^{(0)}$, forming the weights and solving the least squares problem with $\sqrt{w}\ y$ and $\sqrt{w}\ f(\theta)$ as data and model, making the obvious change if there is a weighted nonlinear least-squares routine available.

We now ask - should we modify w and s as we go along and if so how? Since starting values in nonlinear problems are generally not good, we feel that $w^{(0)}$ and $s^{(0)}$ are crude and will need iteration. It is not at all clear how these changes at each step will interact with a specialized algorithm like that of Marquardt. Using this routine with its special start-up procedures to do each step after computing new weights will not be very successful. Direct intervention in the algorithm is required, it cannot just be called each time. Clearly such modifications can be made but we should note that Chambers [1973, p. 7] indicates that such iterative procedures may be inferior to general optimization methods.

5. CREATING SPECIALIZED ALGORITHMS

If one is willing to intervene more directly in an optimization algorithm then some special things can be done to accomodate reweighting and scaling. We shall discuss our efforts in the context of a particular algorithm.

In the past year, work at the NBER Computer Research Center has created a need for nonlinear optimization in such diverse areas as full information maximum likelihood estimation, probit analysis, and projection pursuit [Friedman and Tukey (1974)]. The first algorithm implemented was DOGLEGF, developed by Chien and Dennis at Cornell. This algorithm only requires information about the function F and is closely related to the MINFA algorithm of Powell [1970] which, however, requires the gradient as well as F. DOGLEGF was installed in the NBER TROLL system as a function and is not easily modified except by experienced programmers.

In the TROLL system we also had a symbolic differentiator and a proposed way to automatically compile F,g, and H into very efficient code suitable for repeated evaluation. We also have a macro language that allows a user to glue together various TROLL commands and functions in a way that makes it easy to experiment with new algorithms. With these ideas in mind one of us (RAB) in consultation with John Dennis developed the DOGLEGX algorithm and macro. Since this algorithm formed the basis for further research (by REW) on robust nonlinear regression, we will describe it in detail.

DOGLEGX utilizes a combination of steepest-descent and Newton steps in the process of minimizing a function. As long as gradient steps are relatively large, they are used. However, since gradient steps tend to perform poorly in valleys, Newton steps are also used. Newton steps, however, are of doubtful worth when taken from a point far removed from the minimum. Hence, the algorithm uses a bound on the maximum step size and provides a compromise "dogleg" step which combines the gradient and Newton steps.

As input DOGLEGX requires only the function (all derivatives are computed symbolically), starting values $\theta^{(0)}$, an initial radius (R) to provide an upper bound on step size (the default is zero which makes the first step a gradient step), the maximum number of iterations, and convergence tolerances for the gradient and relative coefficient change.

Initially the expressions for F, g, and H are evaluated. $H(\theta)$ is then forced to be positive definite by the use of a Greenstadt modification. This procedure is carried out whenever the second derivative matrix is reevaluated.

At the beginning of each iteration, there is a test for convergence using both the gradient and the relative change in θ from the previous iteration. The exact details will not be provided here. Assuming that there were no convergence, the algorithm investigates a step in the direction of the gradient vector. Define

$$A^k(\delta) = F(\theta^{(k)})+(g(\theta^{(k)}),\delta)+\frac{1}{2}(\delta,H(\theta^{(k)})\delta)$$

where (a,b) denotes an inner product. The function $A^k(\delta)$ then is a quadratic approximation to $F(\theta^{(k)}+\delta)$ based on the gradient vector and the Hessian. Powell [1970] shows that A^k is minimized along the gradient direction by a step of length

$$\Delta_G = \frac{||g(\theta^{(k)})||^3}{(g(\theta^{(k)}),\ H(\theta^{(k)})g(\theta^{(k)}))}$$

At this point, the step-bound limitation is checked. If $\Delta_G \geq R$ then a step in the gradient direction of length R will be tried. Let δ_N represent the Newton step. If $\Delta_G<R$ and $||\delta_N|| \leq R$ the Newton step is attempted. If $\Delta_G<R$ and $||\delta_N||>R$ a "dogleg" step δ_D is attempted. The dogleg step δ_D is defined as the point on the line connecting δ_G (the gradient step) and δ_N which is at a distance R from $\theta^{(k)}$.

At this point let $\delta^{(k)}$ represent the step that the algorithm decided to take (gradient, Newton, or dogleg). If $F(\theta^{(k)}+\delta^{(k)})<F(\theta^{(k)})$, the step is accepted and we set $\theta^{(k+1)}=\theta^{(k)}+\delta^{(k)}$, otherwise set $\theta^{(k+1)}=\theta^{(k)}$ halve the radius, R, and start a new iteration.

One of the most powerful features of DOGLEGX involves revision of the step bound R. If the step is accepted, a test of the approximation $A^k(\delta^{(k)})$ is performed. If the predicted reduction measured by $F(\theta^{(k)})-A^k(\delta^{(k)})$ is more than ten times the actual reduction, $F(\theta^{(k)})-F(\theta^{(k)}+\delta^{(k)})$, the radius is halved and a new iteration begins.

If this test is passed we perform further checks to decide if the step bound should be increased. In order to do this we look at the scalar product $S(\lambda)=(g(\theta^{(k)}+\lambda\delta^{(k)}),\delta^{(k)})$. The term $\theta^{(k)}+\lambda\delta^{(k)}$ defines a line from $\theta^{(k)}$ in the direction $\delta^{(k)}$. $S(\lambda)$ measures the expected change in the objective function starting at the point $\theta^{(k)}+\lambda\delta^{(k)}$ and taking a step $\delta^{(k)}$. We would like this change to be negative, decreasing the value of the objective function.

At this point we compute $g(\theta^{(k)}+\delta^{(k)})$ so that we have S(0) and S(1) available. If we assume $S(\lambda)$ is linear, these two points define a line and we let λ^* be the point where $S(\lambda)=0$ i.e.

$$\lambda^* = \frac{S(0)}{S(0)-S(1)}$$

If the slope of this line is negative, then $\lambda^*<0$. If the slope is positive, $\lambda^*>0$. When $\lambda^*<0$ or $\lambda^*\geq 2$ a step of twice the length of $\delta^{(k)}$ would still have decreased the value of the objective function if $S(\lambda)$ really were linear. In these cases R is doubled.

If $0\leq\lambda^*<2$ one more check is performed. The predicted gradient at $\theta^{(k)}+\delta^{(k)}$, $g(\theta^{(k)})+H(\theta^{(k)})\delta^{(k)}$, is compared with the actual gradient $g(\theta^{(k)}+\delta^{(k)})$ and if

$$||g(\theta^{(k)}+\delta^{(k)})-g(\theta^{(k)})-H(\theta^{(k)})\delta^{(k)}||^2 \leq .25\ ||g(\theta^{(k)})||^2$$

the step bound, R, is doubled. In all other cases the step bound remains the same for the next iteration. Iterations continue until convergence is reached or the limit on the number of iterations is exceeded.

DOGLEGX was used for testing the ideas developed in section 3. It is an algorithm that invites tinkering (ellipsoids instead of spheres for the step bounding, quadruple instead of double the radius, etc.) and the macro (interpretive) form has permitted this kind of modification, often for specialized purposes. In particular it permitted us to experiment with a number of ideas for robust nonlinear regression.

6. ROBUST NONLINEAR REGRESSION

Since DOGLEGX computes the true Hessian we had no need (at this point) for reweighting as a way to solve our problem. We did however have to consider rescaling and the DOGLEGXS macro was developed by one of us (REW) to accomplish this.

DOGLEGX is complicated by the fact that after it has found an acceptable step it looks ahead at the new gradient to see if it should increase the step bound radius for the next iteration. The question arises - if we are changing the scale, at what point in a step do we change it? Our discussion of this problem is meant to be indicative of the kind of problems that can arise in modifying nonlinear optimization algorithms for specialized applications like robust regression.

In DOGLEGXS we use $\theta^{(k)}$ to compute a scale

$$s^{(k)} = \underset{i}{\text{median}}\ (|r_i(\theta^{(k)})|)/.6745\ .$$

Sections 7 and 8 contain a discussion of starting values and other ways to compute s. The algorithm proceeds as in DOGLEGX until $\delta^{(k)}$ has been determined. Still using $s^{(k)}$, $F(\theta^{(k)}+\delta^{(k)})$ is evaluated and checked to see if $\delta^{(k)}$ is an acceptable step. If the step is not accepted $s^{(k+1)}=s^{(k)}$. If it is accepted we do not yet change s. The test of the approximation $A^k(\delta^{(k)})$ is performed as before. Thus in cases where the radius can be reduced we do not change s before performing these tests. This costs us an extra evaluation of F (we shall

have to eventually evaluate it with a new s) but it is conservative in the sense that changing s here (s generally decreases) would cause us to more often reduce the radius. Reducing the radius is costly because to increase it again we must compute a new g and H, but if a step is not acceptable, no new g and H are necessary to reduce the radius.

If the step has been accepted, we now compute $s^{(k+1)}$ and proceed to see if the radius should be increased assuming, of course, that the test using $A^k(\theta^{(k)})$ was passed (i.e. the radius was not reduced).

The tests for radius increase are the same as before but the new gradient at $\theta^{(k+1)}$ is computed with $s^{(k+1)}$. A number of tests run using $s^{(k)}$ instead of $s^{(k+1)}$ here gave no indication of being better or worse, but were much more expensive since the gradient had to be evaluated twice (with $s^{(k)}$ and $s^{(k+1)}$). The next iteration begins using $\theta^{(k+1)}$ and $s^{(k+1)}$.

7. STARTING VALUES

How to start a robust nonlinear regression is not an easy problem. A scale free start would be nice but least-squares is the only readily available one and, of course, requires a start itself. (Perhaps an L_p, $1<p<2$, start would work, but we have not tried it.) We could also linearize the problem at the supplied starting values and then use least absolute residuals to get a revised start.

We have often found that the original starting values specified by an intelligent model builder can be used directly in a robust loss function with c chosen so that the asymptotic efficiency at the Gaussian is say, .8, i.e.

$$\frac{\left[\int_{-\infty}^{\infty} \rho_c''(t)\, d\Phi(t)\right]^2}{\int_{-\infty}^{\infty} [\rho_c'(t)]^2\, d\Phi(t)} = .8$$

(See Huber (1973) for a discussion of asymptotic efficiency.) For (2.2) this means c is about .67 and for (2.3) about 1. Too low a value of c can throw away a lot of data (low weights) if the start is poor and too high a c does not downweight large residuals enough. We see little reason to perform a least-squares analysis first, although we may want to do this at some point in studying the data.

8. SCALE COMPUTATION

We have used a median absolute deviation (MAD) scaling adjusted so that it will be unbiased for independent Gaussian residuals. In order to allow for a more asymmetric set of residuals, reduce the "granularity" of the median, and remove from the scale computation very large residuals we also tried

$$s^{(k)} = \frac{\sum_{i=1}^{n} w_i^{(k)} r_i^2(\theta^{(k)})}{\sum_{i=1}^{n} w_i^{(k)}}$$

It has performed satisfactorily but requires some form of nonweighted starting scale because $w^{(0)}$ is not defined. All of the results reported below use the MAD scale.

9. CONFIDENCE REGIONS

Since there is not yet general agreement about how to compute covariances for the estimated coefficients in robust linear regression, we cannot hope to give very definitive results for the nonlinear case. Gross (1973) has proposed a way to find confidence intervals for robust location estimates. A partially completed Monte Carlo study by Paul Holland, David Hoaglin, and Roy Welsch indicates that a reasonable covariance estimate for robust linear regression would be

$$\frac{1}{n-p} \sum_{i=1}^{n} w_i\, r_i^2(\hat{\theta})(X^T w X)^{-1} \qquad (9.1)$$

where the w_i are the weights used to obtain $\hat{\theta}$ in the final iteration of reweighted least-squares. The associated t-statistics would probably be based on an equivalent number of degrees of freedom like $\sum_{i=1}^{n} w_i - p$.

An obvious extension to the nonlinear problem is

$$\frac{1}{n-p} \sum_{i=1}^{n} w_i\, r_i^2(\hat{\theta})\, (J^T w J)^{-1} \qquad (9.2)$$

where J and w were used to obtain $\hat{\theta}$. This, of course, has been used in most weighted nonlinear least-squares programs where the weights are assumed to be fixed.

It is useful to see what type of covariance formula arises if we attack the nonlinear problem directly. To do this we follow Bard (1974, p. 176) and argue that we want to examine the effect on the solution θ^* of perturbations in the residuals at θ^*. Bard gives the approximate covariance (in our notation) as

$$V_\theta \doteq \frac{1}{s^4} H^{-1} J^T w V_r w J\, H^{-1} \qquad (9.3)$$

where V_r is the "covariance" matrix of the residuals at θ^* and we have replaced ρ'' by w. One estimate of V_r would be $r(\theta^*)r^T(\theta^*)$. Various other formulas are possible and some have been explored by Tukey (1973). Until more information is available, we prefer to take the approach that (9.3) is conditioned on the weights, set $V_r = \sigma^2 w^{-1}$ and estimate σ^2 by

$$\frac{1}{n-p} \sum_{i=1}^{n} w_i r_i^2(\theta^*) \qquad (9.4)$$

In cases where we ignore the first term of the Hessian, (9.3) would then reduce to (9.2). We have mainly relied on (9.2) especially because robust loss functions tend to reduce the size of the first term of the Hessian (cf. section 4).

10. ELIMINATING SECOND DERIVATIVES

Computing the exact Hessian is expensive even in sophisticated systems. After the above algorithms were developed using DOGLEGX as a base we replaced the exact Hessian by J^TwJ, i.e. we used a type of reweighted least-squares within the context of the DOGLEGX algorithm with scaling. (Call this algorithm DOGLEGW.)

However, as one might expect, this modified algorithm does not work well on some types of highly nonlinear problems. A compromise algorithm (DOGLEGH) is now being tested by Dennis and Welsch. In it, each of the two parts of the Hessian is treated separately. The second part is always computed exactly (except for the fact that w replaces ρ"). The first part is approximated and updated using methods developed by Broyden [see Dennis, (1973)] to update the entire Hessian in general optimization algorithms. This can be accomplished in a way that keeps the Hessian positive definite, removing the need for the Greenstadt modification in DOGLEGX.

11. EXAMPLES

The above algorithms have been tested on a number of standard problems, but we present here an example from marketing which arose in joint work with John Little. The model we are trying to calibrate is

$$SALES(t) = R_o \cdot STREND(t) \cdot PROM.MOD(t) \cdot ADV.MOD(t)$$

$$STREND(t) = SEASON(t) \cdot TREND(t)$$

$$PROM.MOD(t) = 1+B_1 \cdot PROM(t)-B_2 \cdot PROM(t-1)$$

$$ADV.MOD(t) = \sum_{i=1}^{3} \alpha_i \left[C_1 + \frac{C_2 (K \cdot ADV(t-i+1))^{\gamma}}{1+(K \cdot ADV(t-i+1))^{\gamma}} \right]$$

with $\alpha_1 = .46$ $\quad K = .0041$

$\alpha_2 = .32$ $\quad C_1 = .88$

$\alpha_3 = .22$ $\quad C_2 = .24$

and starting values of what we want to estimate

$R_o = 538$ $\quad B_1 = 1$

$\gamma = 2$ $\quad B_2 = .2$

All estimation was done on the first twenty-four observations of the data in Table 1.

Using the loss function of type (2.3) and DOGLEGXS we started the series of computations with c=1 using the given starting values, and then used the results at c=1 to start the computation for c=.8 and c=1.5 corresponding to asymptotic efficiencies of about 50 percent and 95 percent at the Gaussian. The standard errors (using 9.2) are given below the coefficient estimates in Table 2. Also listed are the final value of the (adjusted) MAD scale (s), the weighted least-squares scale (ws) as given in (9.4), the number of evaluations of F, g, and H, the "corrected" degrees of freedom ($\sum_{i=1}^{n} w_i$-p) and the regular degrees of freedom (n-p).

We note that γ is highly sensitive to changes in c and further investigation is called for, including, perhaps, a change in model formulation. The least-squares results are not listed because the algorithm forced γ to infinity (machine overflow) in that case. A more detailed discussion of the model is contained in Little and Welsch [1975].

In order to show how the DOGLEGXS algorithm performed on synthetic data we used the function [see Chambers (1973)]

$$y = e^{-\theta_1 x} - e^{-\theta_2 x} + \text{error}$$

where θ_1 and θ_2 had true values of 1 and 10, ten observations were taken for x=.1(.1)1, and the error was contaminated Gaussian with 75% from N(0,.1) and 25% from N(0,1). The convergence criterion consisted of having the length of the gradient less than .1 and the maximum relative coefficient change less than .001.

All computations were started at θ_1=0 and θ_2=0. The results are listed in Table 3.

12. CONCLUDING REMARKS

We hope that the above discussion will stimulate statisticians to consider the types of algorithms they would like to see developed for a flexible nonlinear fitting package which would include robust loss functions. We also hope that numerical analysts will consider the problems that arise in this area, including large residuals, weights, and the role of special parameters such as scale.

13. REFERENCES

1. Andrews, D.F. (1974). A Robust Method for Multiple Linear Regression. Technometrics 16, 523-531.
2. Bard, Y. (1974). Nonlinear Parameter Estimation. Academic Press, New York.
3. Barrodale, I. and F.D.K. Roberts (1973). An Improved Algorithm for Discrete L_1 Approximation. SIAM J. Numer. Anal. 10, 839-848.
4. Beaton, A.E. and J.W. Tukey (1974). The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data. Technometric 16, 147-192.
5. Chambers, J.M. (1973). Fitting Nonlinear Models: Numerical Techniques. Biometrika 60 1-13.
6. Dennis, J.E. (1973). Some Computational Techniques for the Nonlinear Least Squares Problem, in Bryne and Hall, eds. Numerical Solutions of Systems of Nonlinear Algebraic Equations. Academic Press, New York, 157-183.
7. Gross, A.M. (1973). A Robust Confidence Interval for Location for Symmetric Long-tailed Distributions. Proc. Nat. Acad. Sci. 70, 1995-1997

8. Friedman, J.H. and J.W. Tukey (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis. IEEE Transactions on Computers C-23 9 881-890.

9. Huber, P.J. (1964). Robust Estimation of a Location Parameter. Ann. Math. Statist. 35, 73-101.

10. Huber, P.J. (1973). Robust Regression: Asymptotics, Conjectures, and Monte Carlo. Annals of Statistics 1, 799-821.

11. Huber, P.J. and R. Dutter (1974). Numerical Solution of Robust Regression Problems. Research Report No. 3. Fachgruppe Fuer Statistik.

12. Little, J.D.C. and R.E. Welsch (1975). Robust Calibration of Nonlinear Marketing Models. Unpublished manuscript. Sloan School of Management, M.I.T. Cambridge, Mass.

13. Marquardt, D.W. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. J. Soc. Indust. Appl. Math. 11, 431-441.

14. Powell, M.J.D. (1970). A New Algorithm for Unconstrained Optimization. J.B. Rosen, O.L. Mangasarian, and K. Ritter Eds., Nonlinear Programming, Academic Press, New York, 31-65.

15. TROLL Experimental Programs, Robust and Ridge Regression, NBER Computer Research Center Documentation Series D0070.

16. Tukey, J.W. (1973). A Way Forward for Robust Regression. Unpublished memorandum, Bell Laboratories (Murray Hill).

TABLE 1

MARKETING MODEL DATA

ROW	SALES	PROM	ADV	STREND
1	677.475	1.	750.12	0.95285
2	407.716	0.	118.44	0.999951
3	676.695	0.	507.60	1.01505
4	418.784	0.	90.24	1.01806
5	529.228	0.	81.78	1.03375
6	960.094	1.	902.40	1.13322
7	450.273	0.	98.70	1.10791
8	508.651	0.	177.66	1.08965
9	872.330	1.	454.02	1.09695
10	354.248	0.	14.10	1.05676
11	406.859	0.	45.12	0.897005
12	403.507	0.	177.66	0.83928
13	673.500	0.66	397.62	0.99085
14	483.164	0.34	115.62	1.03967
15	518.784	0.	138.18	1.05525
16	437.880	0.	129.72	1.05826
17	554.404	0.	394.80	1.09082
18	861.341	1.	640.14	1.17766
19	468.277	0.	149.46	1.15123
20	568.979	0.	200.22	1.13209
21	800.701	1.	239.70	1.13955
22	404.365	0.	0.001	1.09768
23	418.706	0.	81.78	0.931605
24	394.388	0.	0.001	0.87156
25	903.819	1.	530.16	1.02885
26	391.426	0.	121.26	1.07939
27	488.230	0.	290.46	1.09545
28	522.915	0.	177.66	1.09846
29	974.980	1.	679.62	1.13210
30	450.896	0.	245.34	1.22210
31	589.634	0.	104.34	1.19455
32	561.029	0.	112.80	1.17453
33	592.673	0.	115.62	1.18215
34	896.882	1.	121.26	1.13860
35	379.735	0.	138.18	0.966205
36	414.965	0.	5.64	0.90384

TABLE 2

MARKETING MODEL RESULTS

c	.8	1.	1.5
γ	1.29 (1.23)	.96 (1.34)	6.81 (9.88)
B_1	.44 (.06)	.53 (.07)	.48 (.06)
B_2	.22 (.03)	.21 (.04)	.23 (.04)
R_o	499.	491.	514.
s	40.8	48.3	39.6
ws	27.2	37.5	44.5
#F	13.	24.	16.
#GH	13.	20.	16.
c.d.f.	12.5	14.6	15.6
d.f.	18.	18.	18.

TABLE 3

TEST FUNCTION RESULTS

c	.8	1.	1.5	LS
θ_1	.75 (.08)	.76 (.09)	.84 (.19)	2.16 (1.19)
θ_2	8.17 (1.17)	8.02 (2.65)	6.78 (2.09)	18.49 (28.85)
s	.15	.15	.21	.52
ws	.06	.07	.14	.95
#F	12.	18.	11.	16.
#GH	12.	17.	10.	16.
c.d.f.	3.6	3.8	4.68	8.
d.f.	8.	8.	8.	8.