(42) Brunvoll, J.; Cyvin, B. N.; Cyvin, S. J. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 171.
(43) Cyvin, S. J.; Bergan, J. L.; Cyvin, B. N. *Acta. Chim. Hung.* **1987**, *124*, 691.
(44) Cyvin, S. J.; Brunvoll, J.; Cyvin, B. N. *MATCH* **1988**, *23*, 189.
(45) Cyvin, S. J.; Brunvoll, J.; Cyvin, B. N. *THEOCHEM* **1988**, *180*, 329.
(46) Cyvin, S. J.; Cyvin, B. N.; Brunvoll, J. *THEOCHEM* **1987**, *151*, 271.
(47) Dias, J. R. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 124.

Application of Microcomputer-Based Robust Regression Methods to Nonlinear Data Analysis

BRETT J. STANLEY and DAVID B. MARSHALL*
Department of Chemistry and Biochemistry, Utah State University, Logan, Utah 84322-0300

Received March 15, 1989

Two nonlinear robust regression methods are developed, characterized, and compared for the example case of exponential decay. Extension to other nonlinear models is obvious. The methods are computationally simple and easily programmed. The resulting programs exhibit reasonable computation times on microcomputers and are thus an easily implemented analytical tool. Programming examples of the robust regression and other methods are given for GAUSS, a particularly convenient language for algorithms that make extensive use of matrix manipulations.

INTRODUCTION

In the analysis of nonlinear data in chemistry, a common procedure is to "linearize" the data and then perform a least-squares fit. In the presence of noise this method can incur significant error because fluctuations in small numbers result in large fluctuations in the natural logs of these numbers (see Figure 1). Furthermore, this method does nothing to alleviate the effect of any outliers that may be present in the data. Robust regression, a technique for the elimination of outlying data points without the need for outlier diagnostic testing, unfortunately does not perform well on the linearized form of such data. Even more, performing data analysis on linearized data is not "correct" in that the least-squares method, which minimizes the sum square of residuals, is not minimizing the "true" model function but some other function, and the minimization itself cannot be guaranteed to have the same transform properties as the original data have.

To avoid these difficulties of analyzing transformed data, nonlinear least-squares analysis is commonly performed. This technique takes the model equation and finds the minimum on the error surface formed by the data and the model. There are many different algorithms available to find this minimum, some of which solve sets of m nonlinear equations, where m is the number of parameters in the model (such as the Gauss–Newton algorithm[1]), many of which find the minimum of a function in m variables (grid and gradient search methods,[2] simplex method,[3] etc.), and methods such as the Levenberg–Marquardt algorithm[4] compromise between downhill gradient methods and the Gauss–Newton step. Although most of these algorithms are fairly rugged in terms of resistance to initial guesses, convergence, etc., parameter estimates can be distorted by "bad" data points, or outliers, just as in the linear case. Other robust methods for nonlinear analysis have been suggested,[5] but are presently quite theoretical in nature, and the corresponding algorithms have not been fully developed or characterized yet for typical data sets that can occur in chemistry.

The nonlinear case presents a much more difficult general solution problem than that of the case of the linear problem which can be set up as the solution of a set of m simultaneous equations. Moreover, the error surfaces (χ^2, or the squared residuals) of exponential data with a corresponding function tend to possess many broad and shallow minima, sometimes making minimization tedious. For quick data analysis in the laboratory, higher level sophisticated algorithms are not applicable for routine execution on microcomputers. Reiterated least-squares solution of maximum likelihood estimators (M-estimators) represents the most general and computationally simplest approach to this problem. In this paper, two nonlinear robust estimators that are easy to program and implement into least-squares analysis routines are derived for the case of exponential data. Specifically, the M-estimators corresponding to least absolute deviations and Tukey's biweight using weighted Gauss–Newton nonlinear least-squares are developed. Extension to other model forms is straightforward. The estimators are tested with the model form in the presence of outliers and various interferences, with and without the presence of Gaussian noise. Comparisons between the two robust methods and between that of ordinary nonlinear least-squares are given, as well as the optimum conditions for the use of one estimator over that of another.

THEORY

A robust estimator is considered "robust" if it is insensitive to the effect of data points that have originated from a parent population that differs from the underlying population being analyzed. There are different types of robust estimators, a particularly common type being the M-estimators. All M-estimators follow from maximum-likelihood arguments in the same sense that least-squares estimates do and are derived as follows. First, a probability distribution to describe the data is chosen, and its maximum is found by minimizing its negative logarithm, ρ.[6] ρ is a symmetrical function of the "scaled" residuals between the data, y_i, and the model, $y(x_i; \theta)$. Minimization is performed by varying the parameter estimates θ

$$\text{minimize over } \theta \quad \sum_{i=1}^{N} \rho(y_i - y(x_i;\theta)/\sigma_i) \tag{1}$$

where σ is the scale and θ is the parameter estimate vector. If we then define the parameter $z \equiv [y_i - y(x_i)]/\sigma_i$, take the derivative of $\rho(z)$

$$\chi(z) \equiv d\rho(z)/dz \tag{2}$$

and set it equal to zero, the generalized normal equations describing the approximate solution at the maximum are then[6]

$$0 = \sum_{i=1}^{N} 1/\sigma_i[\chi(y_i - y(x_i)/\sigma_i)(\delta y(x_i;\theta)/\delta\theta_k)] \quad k = 1, ..., m \tag{3}$$

where m is the number of parameters in the model. In any problem, it is the normal equations that must be solved to obtain the parameter estimates, and the derivative of the model

0095-2338/89/1629-0244$01.50/0

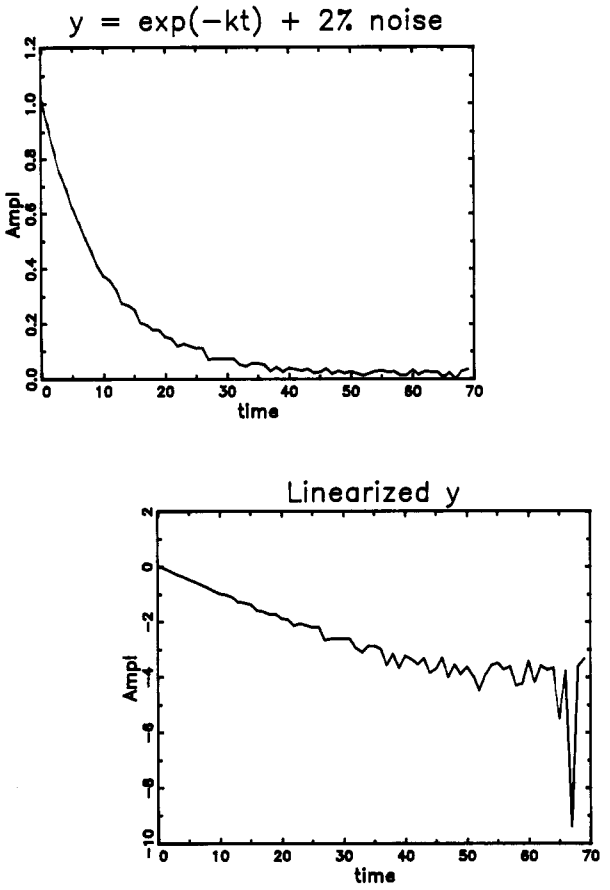

Figure 1. Example of linearization effect: in (a) an example decay is shown containing 2% noise (relative to signal maximum), while (b) shows the same data after linearization. It is clear that the noise is amplified at the end of the signal.

function with respect to the parameter estimates must inevitably be solved. $\chi(z)$ can be considered as a weighting function in these equations. It is the χ functions that distinguish various M-algorithms from one another and are implemented directly into the various iteration schemes that can be used to solve eq 3.

There are other robust methods besides M-estimators that have been characterized, many of which were created to increase the breakdown point, which is defined as the percentage of contamination in the data that the estimator is able to resist and still produce accurate parameter estimates. Generalized M-estimators, or GM estimators, are modifications of the M-estimators that were introduced to resist the effects of outliers in the x direction (or that of the dependent variable—called leverage points); other methods are the L-estimators,[7] R-estimators,[8,9] the resistant line,[10,11] and the median of pairwise slopes,[12,13] all of which have been proposed and characterized for the linear case. Rousseeuw and Leroy have recently published a book[14] on their method of choice, least median of squares (LMS), showing the highest breakdown point possible (50%, including leverage points) for the linear case. For multidimensional data or for a nonlinear application, however, a brute-force method of systematically eliminating all possible permutations of points from the data and calculating the median of the residuals at each permutation is needed. The one permutation with the smallest median is then retained for the scale estimate to use in an M-estimator. Such an approach, although superior, requires extensive computational capacity and is not applicable for use on PC's. We have found that for applications to nonlinear data, an M-method utilizing a reweighted least-squares iteration scheme (using the Gauss–Newton algorithm for nonlinear least-squares) is computationally the most simple and efficient for use with microcomputers. Furthermore, leverage points in time series data such as transient signals can be neglected, for the digitization of the time domain is well characterized.

In reweighted least-squares iteration, the χ function is implemented into a weighting matrix according to[15]

$$w_i = \sum_{i=1}^{N} \chi(y_i - y(x_i;\theta)/s)/(y_i - y(x_i;\theta)/s) \qquad (4)$$

where s is a robust estimate of scale. This weighting matrix is placed into the Gauss–Newton step for nonlinear least-squares

$$\theta^k = \theta^{k-1} + (J_\theta^{k-1\,\mathrm{T}} W J_\theta^{k-1})^{-1} J_\theta^{k-1\,\mathrm{T}} W(y - y(x;\theta)) \qquad (5)$$

where J_θ is the (nxm) gradient or Jacobian matrix of the model with respect to each parameter and T denotes transpose.

In the case of absolute deviations, the χ function is obtained from

$$\rho(x) = |z| \qquad (6)$$

by taking its derivative

$$\chi = \mathrm{sgn}(z) \qquad (7)$$

Substituting into the expression for w_i (eq 4) and rearranging gives the weight for each point in the iteration

$$w_i = s/|r_i| \qquad (8)$$

where r_i is the residual of the ith data point from the corresponding point in the model. For the case of M-estimation using Tukey's biweight

$$\begin{aligned} \chi(x) &= x[1 - (x/a)^2]^2 \quad && \text{for } |x| \le a \\ &= 0 && \text{for } |x| > a \end{aligned} \qquad (9)$$

with the tuning constant $a = 6$. Therefore

$$\begin{aligned} w_i &= (1 - (r_i/(ks))^2)^2 \quad && \text{for } |r_i| < as \\ &= 0 && \text{for } |r_i| \ge as \end{aligned} \qquad (10)$$

The same estimate of scale, s, is used for both procedures. We decided to use the popular median of the absolute values of the residuals,[16] modifying it so only the nonzero deviations are considered:

$$s = \text{median } (|r_i|) \qquad \text{for } |r_i| > 0 \qquad (11)$$

This estimate can be shown to equal 0.67σ.

ALGORITHM

The algorithm used to implement these methods is based on that of Phillips and Eyring[16] (which they call IRLS, iteratively reweighted least-squares) and is as follows:

(1) Take the initial parameter estimates provided by the user and form an initial residual vector. Initial estimates must be fairly good, especially in cases of possible underlying exponentials, but otherwise the requirements for parameter estimates are no more stringent than for ordinary Gauss–Newton least-squares.

(2) Obtain a measure of scale, s, according to eq 11.

(3) Winsorize the residuals. This step is necessary to effectively use step 4, which is to "correct" the initial estimates. Essentially what is done is to chop off the values that the residuals can have at one standard deviation, $\sigma = 1.5s$. That is, residuals less than $-1.5s$ equal $-1.5s$; residuals greater than $1.5s$ equal $1.5s$. Residuals within this window keep their values.

(4) Use the Winsorized residuals in a correction step. Corrections are used to assure that we have the best possible starting guesses before the actual robust iterations begin. This step is

$$q = r/X \qquad (12)$$

and then

$$\theta = \theta + q \tag{13}$$

where q is the correction of θ, r is the Winsorized residual vector, and X is the regressor matrix. For the case of single-exponential decay [i.e., $y = A \exp(-kt) + b$] the natural logarithm of the residuals are used as is an $nx2$ matrix X, where one column is a constant column for the regression of A and b, and the other column equals $-x$, for the regression of the rate constant k. This follows from the linearized model

$$\ln y = \ln K - kt \tag{14}$$

where K contains both the amplitude, A, and the base line, b. It must be remembered to take the exponential of the value of q corresponding to A and b before adding it to θ_A and θ_b.

(5) Obtain a new residual vector with the corrected estimates implemented in the model.

(6) Obtain the scale.

(7) Form the weighting matrix by using eq 8 or 10, where the w_i are contained on the diagonal of W (zeros everywhere else).

(8) Perform weighted nonlinear least-squares by using the Gauss–Newton algorithm and eq 5.

(9) Obtain new residuals.

(10) Test for convergence via the relative change in the parameter estimates being below some convergence criterion (0.001 was used in this work).

$$(\theta^k - \theta^{k-1})/\theta^k < 0.001 \quad ? \tag{15}$$

(11) If step 10 equals "yes", quit; if "no", go to step 6.

EXPERIMENTAL METHODS

All work was done on a 80286/80287 microcomputer (IBM AT "clone") with the programming language GAUSS.[17] We have found this language to be particularly useful in problems in data analysis, owing to its speed and intrinsic capabilities with matrices and classical data reduction procedures. A documented example program written in the GAUSS language is given in the Appendix for the robust method TUKEY, as well as some other illustrative examples common to the field of data analysis, to demonstrate the utility of this language.

Single-exponential decay data were synthesized for the study and modeled by

$$y = A \exp(-kt) + b \tag{16}$$

"True" parameter values were set as $A = 1.0$, $k = 0.1$, and $b = 0.0$, and the number of data points considered was 90. Random noise of zero mean and constant variance, i.e., "white" noise, was added to the data in varying amounts by using a random noise generator; random outliers were added in varying amounts as well by use of the same generator. The "random outliers" are not guaranteed to be outliers (where a subjective criterion for an outlier is that its weight, w_i, approaches zero) because of the nature of the number generator; however, the effect is illustrative of problems that can occur in chemistry (e.g., voltage spikes).

Data interferences were introduced as well. Second-exponential decays were added as were underlying sine waves and Gaussian peaks. The interfering exponential is especially interesting because of its prevalence in experimental chemistry dealing with heterogeneous aggregates of transient signals.

Both the least-absolute deviations variation (LAD) and the Tukey variation (TUKEY) were performed on all data sets along with standard nonlinear least-squares with Gauss–Newton interation (G–N). Comparisons were made and differences in performance were noted. Percent differences in final parameter estimates from their true values are used for the comparison, for variances in the parameter estimates for nonlinear robust regression are not well-defined.

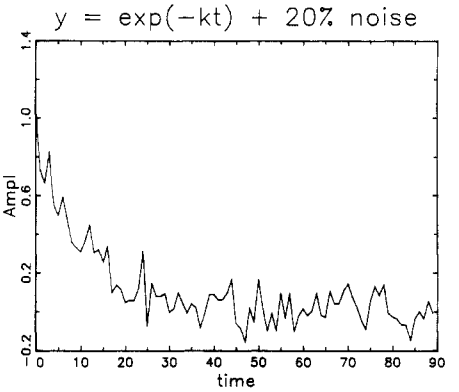

Figure 2. Sample data where the true signal is present with 20% noise.

Table I. Results for Data = Model + White Noise[a]

noise, %	G–N	TUKEY	LAD
1	0.1001	0.1000	0.1002
2	0.1005	0.1004	0.1001
5	0.1015	0.1014	0.1014
10	0.1062	0.1100	0.1103
20	0.09588	0.09697	0.09145
50	0.1146	0.09222	0.1053
100	0.1145	0.1057	0.1024

[a] Percent noise is considered as relative to the maximum signal amplitude.

Table II. Results for Data = Model + x% Outliers

% contamination	G–N	TUKEY	LAD
10	0.1003	0.1000	0.1000
13.3	0.1464	0.1000	0.1000
16.7	0.1841		0.1000
20	0.1538		0.1000
24	0.1909		0.1000
33.3	0.1487		0.1000
50	0.06381		0.08998

RESULTS AND DISCUSSION

The first data set considered is just the model function and added Gaussian noise. A sample decay with 20% noise is shown in Figure 2. The results of all three methods are given in Table I of all noise levels studied. Only the values of the rate constant, k, are presented for clarity and can be considered a good indication of the performance of the estimator. The three methods are not significantly different, as expected. Robust regression cannot be expected to perform any better than least-squares, as least-squares analysis is the optimum estimator in the presence of well-behaved Gaussian noise. But it is comforting to observe that LAD and TUKEY provide comparable results in this case.

Random outliers were introduced in increasing numbers to see what the breakdown points of the estimators would be. We subjectively defined the breakdown point, ϵ, at 0% noise added, as occurring when the estimated parameters "jumped" off the true values. A more rigorous definition of breakdown point would be[14]

$$\epsilon^*(T,x) = \min[m/n;\ \text{bias}(m;\theta,x)] \text{ is infinite} \tag{17}$$

i.e., the smallest fraction of contamination that can cause the estimator T to take on values arbitrarily far from θ.

A sample signal is shown in Figure 3, and the results of the study are given in Table II. Although G–N appears to provide a rate constant estimate in the presence of 10% noise fairly well, it should be considered a fortuitous result, for if the contamination pattern is shifted to the left, so that the first bad point is the first point of the data, G–N breaks down

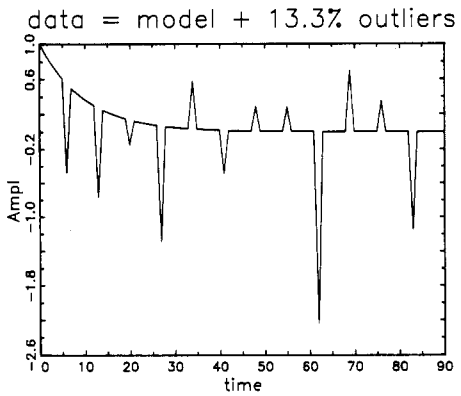

Figure 3. Sample data where the true signal is present with 13.3% of the data points existing as outliers.

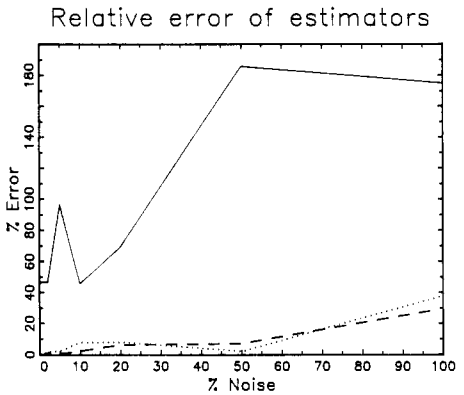

Figure 4. Percent error of the three considered estimators as a function of overlaying white noise when 13.3% of the points exist as outliers. (—) G–N; (- - -) TUKEY; (···) LAD.

Table III. Results for Data = Model + 13% Outliers + x% Noise

noise, %	G–N	% error	TUKEY	% error	LAD	% error
0	0.1464	46.4	0.1000	0	0.1000	0
1	0.1469	46.9	0.1001	0.1	0.1005	0.5
2	0.1469	46.9	0.1012	1.2	0.1019	1.9
5	0.1966	96.6	0.1003	0.3	0.1022	2.2
10	0.1456	45.6	0.1022	2.2	0.1077	7.7
20	0.1694	69.4	0.1062	6.2	0.1078	7.8
50	0.2860	186	0.09287	7.1	0.09801	2.0
100	0.2751	175	0.1296	29.6	0.1379	37.9

miserably. Also, the "randomness" of the outliers caters to least-squares, for if the spikes were all introduced on the positive side of the signal (which could be a realistic occurrence), the result would be total breakdown. In any case, ordinary nonlinear least-squares can be considered to have a breakdown point of approximately 5%. TUKEY provides good estimates up to and including 13.3% contamination, and LAD's breakdown point is seen to be >33.3%. Furthermore, these robust methods hold their estimates fairly well when white noise is added on top of the contamination, as can be seen from Table III and Figure 4, which are the full results at 13.3% contamination.

LAD is seen to be a better method from these results owing to the higher breakdown point and the fact that when TUKEY breaks down, it breaks down by not converging, i.e., no estimate is obtained. We can conclude, then, that LAD's weighting matrix provides for an estimator with better convergence properties than TUKEY's. Indeed, when LAD breaks down, estimates of reduced accuracy (although still accurate compared to G–N) are obtained. The data for the 33.3% contamination–0% noise are shown in Figure 5. It is pleasing to see an estimator handle all that contamination with ease.

When the effect of an interfering exponential was studied, a different characteristic was noted between the two robust

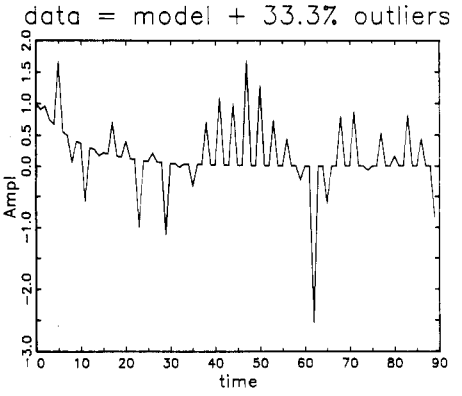

Figure 5. Sample data where 33.3% of the model's data points exist as outliers.

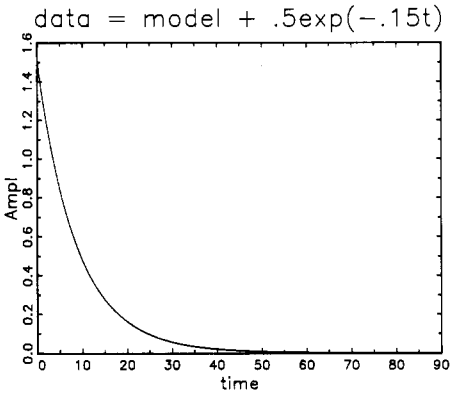

Figure 6. Sample data where a second, interfering exponential is present in addition to that of the model. Interfering exponential follows the law $y = A \exp(-kt) + b$, where $A = 0.5$, $k = 0.15$, and $b = 0$.

Table IV. Results for Data = Model + 0.5 exp(−0.15x)

	k	s	% error
G–N	0.1131		13
TUKEY	0.1033	2.594×10^{-5}	3.3
LAD	0.1122	0.001047	12

methods. Parameter values for this interfering exponential were varied, but following the same exponential law as the data. The data shown in Figure 6 have interfering exponential parameters of $A = 0.5$, $k = 0.15$, and $b = 0$. The results along with the final estimates of scale are given in Table IV. These data are seen to be different from that in the case of distinct outliers, in that the raw data can be fit by the model function fairly well even though a significant interferent is present. Here, advantage is taken of the sensitivity of nonlinear methods toward initial guesses, and one must have a prior idea of what the parameters are supposed to be. As can be seen from the results, TUKEY performs better than LAD under these circumstances. This program seems to find a minimum in between that of least-squares and the true minimum, whereas LAD converges in or around the least-squares area. The reason for this appears to be the more stringent weighting criteria for TUKEY over that of LAD; LAD cannot effectively characterize any part of the curve as outliers, whereas TUKEY maintains a partial handle on the data from the large residuals occurring at the beginning of the signal. This effect is illustrated in the respective estimates of scale. As can be seen from Table IV, s is 40 times larger for LAD than for TUKEY. From Figure 7, which is a residual curve obtained from the TUKEY results, it can be noted that the first 32 points of the data have been weighted out ($w_i = 0$) in this method, which means that TUKEY is essentially fitting the tail of the data only. In the presence of noise it was observed that TUKEY reduced to the effectiveness of LAD and G–N; and for interfering exponentials possessing

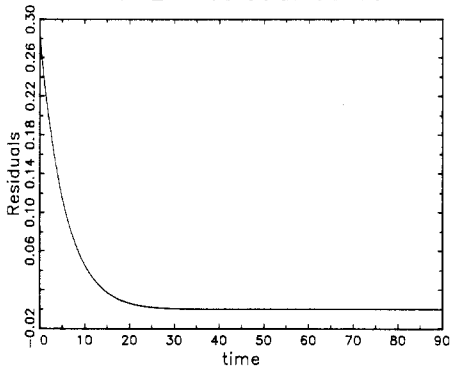

Figure 7. Residual curve resulting from the data of Figure 6 and the curve predicted by the TUKEY method. The interfering exponential is seen to be partially weighted out (A = 0.28 vs A 0.5 total).

larger differences in rate constants from that of the model, the improvement was proportional to the increase in the time scale between the two exponentials.

Other data sets containing various other forms of interferences were studied. If a peak was introduced into the data, TUKEY performed the best, followed by LAD, with the G–N estimate of rate constant significantly too high. If constant interferences like underlying sine waves or sloping base line were introduced, no increase in performance was observed for either robust method because of no available handle. The base-line term, b, simply tried to absorb the effect.

CONCLUSIONS

In the analysis of nonlinear data such as exponential decay, robust estimators can provide an additional tool in data acquisition and analysis. Outliers will be effectively eliminated in the fit; interferences can be exploited as well. All one has to do is to apply the procedures and examine the residual curves to get a picture of any contamination of the data and then act accordingly. One may consider method refinement, or one may just want a rough estimate of the parameters for a model. In any case, it is beneficial to apply all three estimators to suspicious data; their comparisons (of estimates and residual curves) can be enlightening as to the nature of data contamination. Specifically, if one or both robust estimators yield results significantly different from that of G–N, attention should be given to this difference as it most likely results from some form of contamination.

The estimators treated in this paper are computationally simple and can be easily programmed and implemented for analysis on microcomputers. LAD exhibits superiority in the event of outlying points in the data as far as breakdown point is considered; TUKEY exhibits a stronger weighting dependence and is seen to perform better in cases of interferent waveforms. Such estimators provide a convenient addition to the repertoire of data reduction techniques available to the scientist.

ACKNOWLEDGMENT

This work was supported by NSF Grant CHE-8719266.

APPENDIX: THE PROGRAMMING LANGUAGE GAUSS

GAUSS is a multilevel language in which the matrix is the basic unit of numerics instead of the scalar, utilizing numerical algorithms such as those found in LINPACK and EISPACK. Consequently, the length of code needed to implement procedures and programs for numerical analyses with all the linear algebra normally encountered is greatly reduced. The GAUSS programming language is equipped with many functions and procedures that perform statistical calculations and data transformations (such as Fourier transforms and convolutions) and are instituted as an internal part of the language, executable by simple commands. Other such procedures include solving sets of simultaneous equations (linear and nonlinear), procedures for general nonlinear estimation and optimization, and complete internal graphing procedures for "quick" graphics and "publication quality" graphics (2D or 3D).

The most attractive feature of GAUSS besides the large number of prewritten matrix calculation commands, however, is the ability to write your own procedures and incorporate them as a part of the core language. GAUSS is thus easily customized and expanded to suit individual applications. Its execution speed is also quite impressive compared to equivalent compiled FORTRAN routines. FORTRAN and C routines can also be "patched" into and called from within GAUSS procedures or programs. This expandability is a major feature that distinguishes GAUSS from other high-level programming languages that include matrix manipulations as part of the language.

The actual program listings for the robust method TUKEY are shown below. LAD would be the same except the calculation of the weighting matrix. WNLLS is the name of the Gauss–Newton nonlinear least-squares procedure; for ordinary nonlinear least-squares W would consist of diagonal 1's. Note in the following, in particular, the single-line commands for matrix computations—some examples are highlighted in the WNLLS procedure. As mentioned above, another major advantage of GAUSS is that the following procedures can be automatically loaded when GAUSS is started, allowing their use as integral parts of the language. Procedures such as WNLLS can either be called from another procedrue such as TUKEY or function as independent stand-alone procedures. Since GAUSS functions in part as an interpretive language, recompilation to include a procedure in another is not required.

TUKEY.ARC

```
/* Robust Regression for Single-Exponential Decay, written
by B. J. Stanley, 8/10/88.
Current model: y = A*exp(-k*t) + b
Usage:
     {b,s,r}=tukey(y,x,b1,b2,b3,&wnlls);
     y = data vector to be fit
     x = time series corresponding to data
     b1 = initial estimate of amplitude
     b2 = initial estimate of rate constant
     b3 = initial estimate of base line
     &fnlwls = call to Gauss-Newton nonl least-squares, type
in as shown.
     b = parameter estimates provided by wnlls. First number
is amplitude, second is rate constant, third is base line.
     s = final estimate of scale from procedure
     r = final vector of residuals obtained from data and final
parameter estimates. */
proc (3) = tukey(y,x,b1,b2,b3,&wnlls);
  local convg,k,n,w,b,r,sf,s,sfn,i,negx,xr,rl,q,ks,bnew,btest,e,-
nr,rm,rs,rss,nmid,rg,xg,wnlls:proc,rs1;
  convg = 0.001;      /* Set convergence criterion */
  k = 6;      /* Set tuning constant */
  fn fi(b) = b[1,1]* exp(-b[2,1]*x) + b[3,1];      /* Define
model function */
  n = rows(y);
  w = eye(n);
  b = b1|b2|b3;
  r = y-fi(b);      /* Obtain initial vector of residuals */
  gosub scale;      /* Go to subroutine that computes the
estimate of scale, s. */
/* Winsorize residuals */
  sf = 1.5*s;
  sfn = -sf;
```

```
  i = 1;
  do until i > n;
    if r[i,1] < sfn;
      r[i,1] = sfn;
    elseif r[i,1] > sf;
      r[i,1] = sf;
    endif;
      i=i+1;
    endo;
    negx = -xg;      /* Form regressor matrix, xr */
xr = ones(nr,1)~negx;
  rl = ln(rs);      /* Need natural logarithm of residuals
because of linearization */
  q = rl/xr;      /* Correction step */
  q[1,1] = exp(q[1,1]);
  b[1,1] = b[1,1] + q[1,1];
  b[2,1] = b[2,1] + q[2,1];
  b[3,1] = b[3,1] + q[1,1];
  r = y-fi(b);
/* Start iterations */
start:
  gosub scale;
  i = 1;
  ks = k*s;
  w = zeros(n,n);
  do until i > n;      /* Assign Tukey biweights */
    if rsl[i,1] < ks;
      w[i,i] = (1-((r[i,1]/(ks))^2))^2;
    endif;
    i=i+1;
  endo;
  b1=b[1,1]; b2=b[2,1]; b3=b[3,1];
  bnew = wnlls(y,x,w,b1,b2,b3,&fi);      /* With assigned
weights goto weighted nonlinear least-squares procedure and
get new estimates */
  r = y-fi(bnew);
/* Test for convergence */
  btest = (bnew-b)./b;
  btest = abs(btest);
  if btest < convg;
    b=bnew;
    retp(b,s,r);
  endif;
  b=bnew;
  goto start;
/* Subroutine to estimate the scale from the median of the
nonzero absolute deviations */
scale:
  i=1; e=zeros(n,1);
  do until i > n;
    if r[i,1] = = 0;
      e[i,1] = 1;
    endif;
    i=i+1;
  endo;
  rg = delif(r,e); xg = delif(x,e);
  nr = rows(rg);
  rm = nr%2;
  rs = abs(rg); rs1=abs(r);
  rss = sortc(rs,1);
  if rm/=0;
    s = submat(rss,nr/2+.5,0);
  else;
    nmid = nr/2;
    s = meanc(submat(rss,nmid,0)|submat(rss,nmid+1,0));
endif;
  return;
endp;
```

WNNLS.ARC

```
/* Written 8/8/88 by B. J. Stanley. This procedure is a
modified version of NONLIN.PRG for single exponentials, as
described by Lee Edlefsen, 3/18/1985. The program uses the
Gauss-Newton step incorporating a weighting matrix at every
iteration. The analytic derivatives for the exponential are used
to compute the gradient matrix, g, instead of numerical gra-
dients to speed up execution. "Squeezes" are calculated at each
iteration to adjust the step length, db, to its optimal value. This
program minimizes, with respect to the parameter estimates
b, the function S(b) = sum_i(yi - fi(b))^T W(yi - fi(b)), where
fi(b) is currently set to fi(b) = A exp(-kt) + b. */
  proc wnlls(y,x,w,b1,b2,b3,&fi);
  local  b,g,m,u,gu,db,k,n,maxsqez,ssr,b_new,s_db,s_b,s_
b1,s_lm,s_itr,lbl,s_2,lb2,c_b;
  b = b1|b2|b3;
  maxsqez = 10;
  k=3; n=rows(y); db=1;
  do until abs(db) < 1e-5;
/* Note single-line calculations below that would require
do-loops in FORTRAN and other languages. */
  g = zeros(n,k);
  g[.,1] = exp(-b[2,1]*x);
  g[.,2] = -x*b[1,1].*exp(-b[2,1]*x);
  g[.,3] = ones(n,1);
/* Note single-line matrix calculations below that would each
require major subroutines in FORTRAN or other languages.
*/
  m = g'w*g;
  u = y-fi(b);
  gu = g'w*u;
  db = solpd(gu,m);
/*--*/
  gosub squeeze(b,db);
  pop b_new;
  db = b-b_new;
  b = b_new;
endo;
retp(b);
squeeze:
  pop s_db; pop s_b;
  s_b1=s_b + s_db; s_lm=1/2; s_itr=1;
  gosub compl(s_b1); pop lbl;
    do until s_itr > maxsqez;
    s_b2=s_b+s_lm*s_db;
    gosub compl(s_b2); pop lb2;
    if lb1 >= lb2;
      s_b1=s_b2; s_lm=s_lm/2; s_b2=s_b + s_lm*s_
db; lbl=lb2;
  s_itr=s_itr+1;
    else;
    return(s_b1);
  endif;
endo;
return(s_b2);
compl:
  pop c_b;
  u = y-fi(c_b);
  ssr = u'*w*u;
  return(ssr);
endp;
```

OTHER PROGRAMMING EXAMPLES

The examples given below illustrate the very compact form of many GAUSS expressions for matrix computations. In example 1, note the compact way in which the real and im-

aginary parts of the fft result are stripped out and combined to yield the spectral amplitude. [The "fft procedure" is called by issuing a single command, fft(x).] Example 2 illustrates the compact way in which means and maximums of data are calculated. Example 3 shows the calculations of the nonlinear maximum entropy spectrum from interferometric data using the Burg algorithm.[18] It also illustrates the compact form of matrix computations and the combination of a FORTRAN subroutine with a GAUSS procedure.

Example 1: FTSPEC.ARC

```
/* Written 4/26/88 by D. B. Marshall. Computes and displays the spectrum (frequency-space) from a Fourier transform of time-space data, using spectrum = sqrt(real^2 + imaginary^2).
Usage: s = ftspec(&graphx,f)
  &      Call to graphing procedure. Type in as shown.
  f    FT matrix, result of using fft procedure. Contains real part in column one, imaginary part in column 2.
  s      Output, spectral amplitude vector. */
proc ftspec(&gx,f);
  local d,r,i,x,xr,s,gx:proc;
  r=submat(f,0,1); i=submat(f,0,2);
  xr=rows(r); x=seqa(0,1,xr);
  s=sqrt((r^2) + (i^2));
  d=gx(x,s);
  retp(s);
endp;
```

Example 2: NOISE.ARC

```
/* Procedure to calculate random noise with mean = 0, as a percentage of maximum signal amplitude, and add to signal. Written 5/9/88 by D. B. Marshall.
Usage: sn = noise(s,p);
  s      signal to which noise will be added
  p      % of noise (rel. to max. signal amplitude)
  sn     output, signal with noise added */
proc noise(s,p);
  local n,c,rn,mrn,sns,ms,sn;
  n=rows(s); c=cols(s);
  ms=maxc(s); sns=(p/100).*ms;
  rn=rndn(n,c); mrn=maxc(abs(rn));
  rn=rn./mrn; rn=rn-meanc(rn)';
  rn=rn.*sns;
  sn=s+rn;
  retp(sn);
endp;
```

Example 3: NLMEM.ARC

```
/* Procedure to do nonlinear maximum entropy calculations of time series data. Calls BURG.GXE to calculate prediction error filter coefficients and the filter power. (BURG.GXE is compiled and linked GAUSS-executable FORTRAN code: see BURG.FOR for source code.) Algorithm based on the Burg method for calculating maximum entropy spectra. Written 5/10/88 by D. B. Marshall.
Usage: s = nlmem(x,lp,r,&fft);
  x      time series data of n points, n=1024 max.
  lp     length of prediction error filter: should be between n/3 and n/2.
  r      exponent of spectral resolution (=2^r), max 10.
  &fft     call to fft as procedure - type as shown.
  s      output, maximum entropy spectrum */
proc nlmem(x,lp,r,&ft);
  local ft:proc,lfile,s,n,fp,rfp,ifp,sfp,p,v,hn;
  lfile="burg.gxe";      /* specify burg code file */
  local burg;
  burg=zeros(4131,1);     /* reserve space for BURG.GXE */
  p=zeros(1024,1); v=0;      /* initialize BURG.GXE output */
  n=rows(x); hn=n/2;
  if lp > hn;
  lp=hn; endif;     /* limit lp and r to avoid BURG.GXE crash */
  if r > 10;
  r=10; endif;
  x=x';     /* transpose data to FORTRAN convention */
  loadexe burg=^lfile;
  callexe burg(n,x,lp,r,p,v);       /* call FORTRAN code BURG.GXE */
  x=x';       /* transpose back to GAUSS convention */
/* calculate FT spectral amplitude */
  fp=ft(p);
  rfp=submat(fp,0,1); ifp=submat(fp,0,2);
  sfp=sqrt((rfp^2) + (ifp^2));
  sfp= (abs(sfp))^2;
/* calculate MAXENT spectrum using Cholesky decomposition */
  s=v/sfp;
  retp(s);
endp;
```

```
      SUBROUTINE BURG(LX,X,LA,M,A,V)
C     NONLINEAR MAXIMUM ENTROPY ROUTINE BASED ON THE BURG
C     ALGORITHM RETURNS VALUES OF PREDICTION ERROR FILTER (A) AND
C     FILTER POWER (V) FROM INPUT OF TIME SERIES (X), LENGTH OF
C     TIME SERIES (LX), LENGTH OF PREDICTION ERROR FILTER (LA),
C     AND INTEGER EXPONENT OF SPECTRAL RESOLUTION (M).
C     NP=2**M CONTROLS SPECTRAL RESOLUTION
C
C     MODIFIED FROM ROUTINE BY S. HAYKIN, "NONLINEAR METHODS OF
C     SPECTRAL ANALYSIS, S. HAYKIN, ED., SPRINGER-VERLAG,
C     BERLIN, 1983, APPENDIX 6, pp. 307-313.
C
      IMPLICIT DOUBLE PRECISION (A-H, L-Z)
      DIMENSION X(1024),F(1024),B(1024),A(1024)
      LA=IDINT(LA)
      LX=IDINT(LX)
      M=IDINT(M)
      NP=2**M
      V=0.
      DO 1 I=1,LX
      V=V+X(I)**2
      (F(I)=X(I)
    1 B(I)=X(I)
      V=V/FLOAT(LX)
      DO 5 N=1,LA
      SN=0.
      SD=0.
      L=N+1
      DO 2 J=L,LX
      SN=SN+F(J)*B(J-1)
    2 SD=SD+F(J)*F(J)+B(J-1)*B(J-1)
      G=2.*SN/SD
      V=V*(1.-G*G)
      A(N)=G
      IF(N.EQ.1) GOTO 4
      NH=N/2
      DO 3 K=1,NH
      KK=N-K
      AA=A(KK)-G*A(K)
      A(K)=A(K)-G*A(KK)
```

```
3   A(KK)=AA
4   M1=N+1
    BS=B(N)
    DO 5 K=M1,LX
    FF=F(K)
    BB=BS
    BS=B(K)
    F(K)=FF-G*BB
5   B(K)=BB-G*FF
    DO 6 I=1,LA
    K=LA-I+1
6   A(K+1)=-A(K)
    A(1)=1.
    LP=LA+2
    DO 8 I=LP,NP
8   A(I)=0.
    RETURN
    END
```

REFERENCES AND NOTES

(1) Norton, J. P. *An Introduction to Identification*; Academic Press: Orlando, FL, 1986.
(2) Bevington, P. R. *Data Reduction and Error Analysis for the Physical Sciences*; McGraw-Hill: New York, 1969.
(3) Nelder, J. A.; Mead, R. *Comput. J.* 1965, *8*, 308.
(4) Wolfe, M. A. *Numerical Methods for Unconstrained Optimization*; Van Nostrand Reinhold: Workingham, U.K., 1978.
(5) Dutter, R.; Humber, P. J. *J. Stat. Comput. Simul.* **1981**, *13*, 79-114.
(6) Press, H. P.; Flannery, B. P.; Teukolsky, S. A.; Vetterling, W. T. *Numerical Recipes*; Cambridge University Press: Cambridge, 1986.
(7) Bickel, P. J. *Ann. Stat.* **1973**, *1*, 597-616.
(8) Jureckovâ, J. *Ann. Stat.* **1977**, *5*, 364-372.
(9) Jaeckel, L. A. *Ann. Math. Stat.* **1972**, *5*, 1449-1458.
(10) Velleman, P. F.; Houglin, D. C. *Applications, Basics, and Computing of Exploratory Data Analysis*; Duxbury Press: Boston, 1981.
(11) Johnstone, I. M.; Velleman, P. F. *J. Am. Stat. Assoc.* **1985**, *80*, 1041-1059.
(12) Adichie, J. N. *Ann. Math. Stat.* **1967**, *38*, 894-904.
(13) Sen, P. K. *J. Am. Stat. Assoc.* **1968**, *63*, 1379-1389.
(14) Rousseeuw, P. J.; Leroy, A. M. *Robust Regression and Outlier Detection*; Wiley: New York, 1987.
(15) Launer, R. L.; Wilkinson, G. N. *Robustness in Statistics*; Academic Press: New York, 1979.
(16) Phillips, G. R.; Eyring, E. M. *Anal. Chem.* **1983**, *55*, 1134-1138.
(17) GAUSS is available from Aptech Systems, Inc., 26250 196th Place South East, Kent, WA 98042 [(206) 631-6679].
(18) Burg, J. P. Paper presented at the NATO Advanced Study Institute on Signal Processing, Enchede, Netherlands, 1968. Also cited in *Nonlinear Methods of Spectral Analysis*; Haykin, S., Ed.; Springer-Verlag: Berlin, 1983; Appendix 6.

Using CONCORD To Construct a Large Database of Three-Dimensional Coordinates from Connection Tables

ANDREW RUSINKO III, ROBERT P. SHERIDAN, RAMASWAMY NILAKANTAN, KEVIN S. HARAKI, NORMAN BAUMAN, and R. VENKATARAGHAVAN*

Medical Research Division, Lederle Laboratories, American Cyanamid Company, Pearl River, New York 10965

Received May 5, 1989

The program CONCORD was used to generate a database of three-dimensional (3D) structures from a large (≈265 000 structures) database of connection tables. Additional chemical information was introduced into the 3D database to facilitate the rapid searching for complex 3D substructures such as pharmacophores. Each non-hydrogen atom was characterized by five properties: element type, number of non-hydrogen neighbors, number of π electrons, number of attached hydrogens, and formal charge. For some common functional groups, the number of hydrogens and the formal charge were assigned by considering the most likely ionization state at physiological pH. Dummy atoms were created to represent centroids of flat rings, perpendiculars to those rings, and lone-pair/proton positions. A three-dimensional structural database of 223 988 Cyanamid structures (CL File) was generated. In addition, a subset of the Cambridge Structural Database of experimentally determined molecular geometries was converted into a similar format suitable for 3D substructure search.

INTRODUCTION

Most chemical and pharmaceutical companies maintain large databases of chemical structures containing, among others, structures of their own proprietary compounds. These structures are almost always represented as connection tables (atoms and bonds) and are searchable by topological substructure search systems (e.g., MACCS[1] or the MEDCHEM[2] system). American Cyanamid, like other companies, maintains a collection of samples corresponding to the compounds in the structural database. In conventional screening programs, biologists or chemists may select subsets of structures from the database, so that a small amount of the corresponding sample can be tested in one or more biological assays. This selection can be random (exploring a structurally diverse set for "active" candidates) or guided by some structure–activity model.

Many techniques in the literature attempt to relate the chemical information contained in connection tables (i.e., a two-dimensional structural diagram) to biological activity.[3-6] However, the more sophisticated "graphical" methods of molecular modeling are based on three-dimensional (3D) structures of molecules, and structure–activity rules derived from such methods are naturally expressed in three dimensions. For example, the object of "active analogue" modeling[7-9] is to deduce a pharmacophore, a hypothetical configuration of chemical groups essential for biological activity. One would then like to know which compounds in a proprietary database contain a given pharmacophore. However, since compounds in the database are stored as connection tables and not 3D structures, this type of question is quite difficult to formulate. To properly address 3D questions, one must transform the database of connection tables into 3D molecular structures.

In this paper we will demonstrate how a database of 3D coordinates can be constructed from a large database of connection tables. In the accompanying paper,[10] we discuss

* Author to whom correspondence should be addressed.

0095-2338/89/1629-0251$01.50/0