Dynamic Graphics and Nonlinear Regression Diagnostics using XLISP-STAT

R. D. Cook, S. Weisberg, and N. Wetzel*

January 15, 1999

Abstract

A nonlinear regression prototype, nonlin-model-prototype in LISP-STAT (Tierney, 1990) is described. This prototype builds on the methods in regression-model-prototype as described in Dynamic Graphics and Regression Diagnostics using XLISP-STAT. The code for these additions is available in the XLISP-STAT library in statlib.

This document is a scratch copy and is thus not completely accurate.

1 Introduction

The LISP-STAT environment designed by Tierney (1990) provides an excellent platform for dynamic graphics in statistics. We have used this program as a basis to implement the ideas for nonlinear regression diagnostics that appear in various papers. The code and this document assume that the reader is familiar with the technical report Dynamic Graphics and Regression Diagnostics using XLISP-STAT, (hereafter called DG+RD) and that the user of the code has already loaded the code referenced in this document. This document describes my code, gives an indication of how the code works, summarizes what the code does, and provides help on getting started. To use the code, you will need to have a copy of XLISP-STAT. In this document, we assume that this program is available to you. We have tested the code on Unix workstations and on Macintosh II computers; we have no reason to believe that it will not work with the Windows version of XLISP-STAT.

We are distributing this code for several reasons. First, we believe that the methods provided here can be very helpful in nonlinear regression modeling. The methods are difficult or impossible using standard programs. Second, we show how XLISP-STAT can be used to implement new statistical ideas. Third, we provide code that may be helpful to others who are interested in developing similar methodology.

These enhancements to the nonlin-model-proto are available without charge, and may be freely distributed or modified. However, they are not to be sold for a profit, or offered as an inducement to purchase a commercial product. In general, this code may be distributed using the same rules for distributing XLISP-STAT. Although we have attempted to produce bug-free code, errors surely remain. We would be glad to hear about bugs; send e-mail to wetzel@umnstat.stat.umn.edu.

1.1 Using statlib To Get the Code

The code will eventually be available in the XLISP-STAT library on statlib. The get the code over the internet, send the message:

mail statlib@lib.stat.cmu.edu
Subject:
send nonlin from xlispsstat

*This work was supported by National Science Foundation Grant number 9001298?????. We would like to thank Luke Tierney for his help with this project.
Save the information that is sent back to you in a file, and follow the instructions included for decoding. When unpacked, you will have a LaTeX copy of this document called nonlinindoc.tex and several files whose names all end in .lisp. Put all of the .lisp files into the directory (or, on the Macintosh, folder) from which you launch XLISP-STAT. This code is an extension of the regression diagnostic methods described in DG+RD, and thus you should put the .lisp files in the directory which has the files from that document.

1.2 Distribution by ftp

The code may be obtained via anonymous ftp to the machine umnstat.stat.umn.edu. Here is an abridged transcript of a session that will get the code:

```
your-machine% ftp umnstat.stat.umn.edu
Connected to umnstat.stat.umn.edu.
Name (umnstat.stat.umn.edu:anonymous): anonymous
Password (umnstat.stat.umn.edu:anonymous): (use your e-mail address)
ftp> cd pub
250 CWD command successful.
ftp> get nonlin.sar
200 PORT command successful.
150 Opening data connection for dyndiag.sar (ascii mode) (201290 bytes).
226 Transfer complete.
local: nonlin.sar remote: nonlin.sar
????????????205792 bytes received in 11 seconds (18 Kbytes/s)
ftp> quit
```

The file nonlin.sar is a Unix shar file, identical to the one that could be obtained via statlib.

1.3 Distribution by Disk

If you have the code on a floppy disk, copy every file on the disk that ends with .lisp into the directory (folder on the Macintosh) from which you launch XLISP-STAT. The remaining file, nonlinindoc.tex is a LaTeX copy of this document.

1.4 Getting Started

When XLISP-STAT starts, it looks for a file called statinit.lisp in the current directory or folder. If such a file exists, it is loaded by the system. To load the dynamic diagnostics code automatically, modify your statinit.lisp file, or create a new one, by adding the following line:

```
(load "winit")
```

To load the code manually, enter the above command in the listener window (which is the name of the window for typing commands or getting output), or, on the Macintosh, select “Load” from the “File” menu, and, in the dialog box that will appear, select “cwwinit.lisp.”

To get a quick overview of the enhancements, enter the following command:

```
(load "clarke")
```

This will create a nonlinear regression object called clarke, and this object will be sent the :graphics menu "Clarke" message. A menu called Clarke will appear in your menu bar. You can try out the items in this menu as a test of the features available.
2 Nonlinear Regression Methods

Nonlin-model-proto is a prototype for fitting nonlinear regression models using LISP-STAT. It is different from the original version, nreg-model-proto, written by Luke Tierney; the major difference being that in the Tierney version the mean function is a function of just the parameters, and in this new version the mean function is a function of the parameters and the data. This prototype inherits from regression-modelproto and thus includes all of the regression methods. In addition, I have included parameter plots (see Cook, 1987), profile plots (see Cook and Weisberg, 1990), projected residuals (see Cook and Tsai, 1985), and Jacobian leverages (see St. Laurent and Cook, 1991).

2.1 Calling sequence

The model is initialized with a function call as described in below. There are four required arguments, pred-vars, y, mean-function, and theta. The mean-function should be a function of theta and x which returns \( f(\theta, x) \). Y is the observed data when \( x = \text{pred-vars} \), and theta is the starting value for the optimization. In addition to the keywords allowed in the regression model, there are six more keywords. These are count-limit, parameter-names, pred-var-names, exact-first-derivative, and exact-second-derivative. These will be described later in this document. Here is an example of the function call, assuming the data is in a file called "clarke.dat" in the current directory:

```
(load "clarke.dat")
(defun clarke (lambda (th x)
  (+ (third th) (* (second th) (exp (* (first th) x))))))

(defun clarke (nonlin-model clarke-x
  clarke-y
  clarke-fun
  (list -1.05 2.55 .98)
  :graphics-menu "Clarke")

The file "clarke.dat" defines the variables clarke-x, and clarke-y. The graphics-menu keyword works exactly as described in Dynamic Graphics and Regression Diagnostics using XLISP-STAT.

To get a profile plot, enter:

```
(send clarke :profile-plot)
```

or select the "Profile Plot" item from the Clarke menu.

The parameter-names, and pred-var-names keywords allow the user to specify the names of the parameters and the pred-vars. These names default to "Parameter 0", "Parameter 1", etc. and "Pred 0", "Pred 1", etc. The count-limit keyword is described in section 2.3, and the exact-derivative keywords are described in section 2.6.

2.2 Routines used

This code uses standard XLISP-STAT routines, plus bits and pieces of ALL the code provided. Thus, you must load EVERYTHING for the code to work properly, including EVERYTHING from the DG+RD document. In addition to the nonlinear regression methods, a number of significant changes have been made to the contour-plot-proto in order to eliminate unnecessary recalculation of the contoured function.

\[^1\text{On Unix systems, the current directory is the same as the directory from which XLISP-STAT is started. On the Macintosh, the current directory or folder is the last folder accessed. For example, if a file in the folder "Datasets" is opened for editing using the built-in editor, then the current folder is "Datasets."}\]
2.3 Computational comments

Nonlin-model proto uses the newtonmax algorithm as implemented in XLISP-STAT to obtain the estimates of the coefficients. The newtonmax algorithm is sent the count-limit keyword and thus this keyword is used in determining convergence. After convergence is obtained, a more accurate first derivative matrix is calculated.

All three-dimensional array operations are done using the square-bracketmult and 3d-matmult functions found in the file nonlin-regr3.ksp.

2.4 Plotting methods

Most of the plots created by sending a message to a nonlin model proto can later be accessed and will be updated whenever the model is updated. In particular, when a case is deleted or restored, the nonlinear regression model is recomputed, and most plots are redrawn. Most plots have three additional menu items added to their standard menu. These menu items, which appear at the end of the menu, can be used to override automatic plot updating or to delete or restore cases to a model from the plot. Points not used in computations will generally appear as selected points on a plot, but of course they can be marked as desired in the usual ways. Where there are points not used in calculations, one of the additional menu items is “Select Deletions”, so these points can always be identified.

Because of the amount of time required to do the nonlinear regression fitting, the user may want to uncheck Plot Updating menu item for plots which demand a significant amount of time to calculate. These plots will then remain un-updated.

In addition, all plots include a number of plot controls, buttons and sliders that can do useful things to the plot. Almost all of these are self explanatory, but see also the section on Plot Controls.

All of the plot methods that are available for linear regression are also available for nonlinear regression and can be made using the same commands as described in Dynamic Graphics and Regression Diagnostics using XLISP-STAT.

The additional types of plots are described below:

- :parameter-plot Q Constructs a parameter plot for the chosen parameter. A parameter plot is useful when doing inference on a selected parameter. The user will be given a dialog box to choose the parameter. The calculations are done as follows: Let $V = UR$ denote the QR-factorization of $V$. Partition $R$ as:

$$R = \begin{bmatrix}
R_{11} & R_{12} \\
0 & R_{22}
\end{bmatrix}$$

where $R_{11}$ is $p-1 \times p-1$. Let $A_2$ denote the last $p \times p$ face of the unscaled parameter effects curvature array (\texttt{:ap}) and partition $A_2$ as:

$$A_2 = \begin{bmatrix}
A_{211} & A_{212} \\
A_{212} & A_{222}
\end{bmatrix}$$

where $A_{211}$ is $p-1 \times p-1$. Finally partition $U = (U_1, U_2)$, where $U_2$ is $n \times 1$. If the chosen parameter is the last parameter, the parameter plot is a three dimensional plot of the ordinary residuals vs $U_2$ and $U_1 A_{212}$. For details of this plot see Cook (1987, Biometrika) and section 3.2 of Cook and Goldberg (1986 Annals of Statistics).

If the chosen parameter is not the last parameter a we obtain the appropriate $A$ and $U$ in the rotated coordinate system. This is done in the following way: Let $Z$ be a permutation matrix permuting the last and selected column. Let $RZ^T = U^* T R^*$ be the QR-factorization of $RZ^T$. Let $U_z = U z$ and $A_z = [U^* [U^* A U^*]^T]$. The $U_z$ and $A_z$ are the $U$ and $A$ arrays in the rotated coordinate system, so use $U_z$ and $A_z$ instead of $U$ and $A$ in the preceding paragraph.
Figure 1: Parameter Plot for gamma in the jet fighter data

The parameter plot is a generalization of the added variable plot from linear regression. Informally, the plot of the residuals versus $U_2$ reflects the main effects that contribute to the estimate of $\theta_2$ (the chosen parameter). And, the plot of the residuals versus $U_1A_{212}$ reflects the interaction between $\theta_2$ and the other parameters.

For example, to reproduce Figures 4 and 5 from Cook (1987) one must first load the jet fighter data:

(load "jet")

This will create a nonlinear regression object jet and will send it the :graphics-menu "Jet" message. One can either type the command

(send jet :parameter-plot)

or select Parameter Plot from the menu. In either case a dialog box appears. In this dialog box, select the parameter Gamma, and click the OK button. This will produce a three dimensional plot, the vertical and horizontal axes reproduce Figure 4 of Cook (1987) and the vertical and out of page axes reproduce Figure 5 of Cook (1987). The reproduction of Figure 4 of Cook (1987) is displayed here in Figure 1.

- :profile-plot (key (perc .95)) Constructs a profile log likelihood plot for the parameter of interest. This can then be used to determine exact confidence intervals for the parameters. The user will be given a dialog box to choose the parameter ($\theta_i$). The horizontal axis represents the $t$-value for the test of $H_0 : \theta_i = \theta_{\text{p}}$ versus $H_1 : \theta = \theta_{\text{i}}$, where the $\theta_{\text{p}}$'s are the values on the vertical axis. The values on the vertical axis begin at $\theta_{\text{i}}$ and are incremented and decremented until the value on the horizontal axis is larger than $t_{1-\text{perc},n-p}$. If an exact first derivative is supplied to the nonlin-model-plot it will be used in the calculation of this plot.
This method is described in detail in Cook and Weisberg (1990), however I have not implemented the enhancements described in section of this paper. I use a step size of \(0.2 \times \text{SE}(\hat{\theta}_i)\) and each starting value is the ending value from the last calculation. Because of the amount of calculation the profile plot is a non-updating plot. Even if you check the Plot Updating item in the plot menu, this plot will not update, because the way the plot updating works assumes that the same number of points will be plotted and this may not be the case with a profile plot. If the model has changed and you want to see the updated profile plot you must go through the plot creation process again.

One of the examples in the above paper is found in section 3 of that paper. In order to reproduce this figure, one must load the clarke data:

```lisp
(load "clarke")
```

This will create a nonlinear regression object clarke and will send it the :graphics-menu "Clarke" message. One can either type

```lisp
(send clarke :profile-plot)
```

or select the Profile Plot item from the menu. In either case a dialog box appears. The three plots in Figure 3 of Cook and Weisberg (1990) will each be produced if one selects Parameter 0, Parameter 1, or Parameter 2 respectively. the Wald intervals may be placed on the plot by clicking on the Wald Curves button once the plot has appeared. The other buttons on this plot will use a linear interpolation of the plotted points to give an exact confidence interval. For example, with the profile plot for Parameter 1 in the clarke example, if one clicks on the 90% Conf. Int. button, horizontal lines will appear at Parameter 1 equals 2.1812 and 3.42365. Also, a message will appear in the upper right hand corner of the plot saying that the 90% confidence interval is (2.1812, 3.42365). This plot is shown in Figure 2.

Figure 2: Profile Plot for parameter 1 in the Clarke data set.
Figure 3: Approximate and exact 95% confidence regions for $\theta_0$ and $\theta_2$

- **2d-contour-plot** (select (num-points 6)) Constructs a two-dimensional profile plot for the parameters of interest. This is used to construct two-dimensional confidence regions. The user will be given dialog boxes to choose the two parameters ($\theta_i$ and $\theta_j$). The value of the function whose contours are shown is the $F$-value for the test of $H_0: \theta_i = \theta_j = \theta_{ij}$ versus $H_1: \theta_i = \theta_i^*, \theta_j = \theta_j^*$, where $\theta_{ip}$ and $\theta_{jq}$ are the points on the horizontal and vertical axes, respectively. This function is calculated on a num-points by num-points grid.

To reproduce Figure 1 of Cook and Witmer (1985, JASA), one must first load the amino acid data:

(load "amino")

This will create a nonlinear regression object amino and will send it the :graphics-menu "Amino" message. On can either type

(send amino :2d-profile-plot)

or select the 2d Profile Plot item from the menu. In either case a dialog box will appear. Select Parameter 0 and then click on OK. Another dialog box will appear. Select Parameter 2 and the click OK. After a lot of calculation, a contour plot will appear. By default the 99% contour is initially shown. To get the 95% contour, click on the Add level(s) box and enter 0.95. In order to remove the 99% confidence curve, click on the Delete level(s) box and select .99 from the dialog box that appears. Finally, click on the Approx. Conf. Reg. box and enter .95 in the dialog box. If you are using a color machine the approximate confidence region will appear in red. The resulting plot (shown in Figure 3) will be similar to Figure 1 of Cook and Witmer (1985). Any differences are due to the grid used to calculate the contours.

The Double Grid and Add To Grid boxes on the contour plot can be used to add horizontal or vertical lines to the grid. For example, the default grid is 6 by 6 (36 function evaluations).
If the user clicks on the Double Grid box, the function grid will become 11 by 11, adding a line in between all existing lines (an additional 75 function evaluations). The Spin Function box can be used to get a 3-d plot of the function.

- :distance-plot (key (x-axis 'case-numbers)) Constructs a plot of cooks-distances versus the chosen x-axis. Notice that the x-axis defaults to be the case-numbers. This plot can then be used to get a plot of the other types of distances. This can be done in the following way. Click and hold on the Dist. Options button of this plot. A menu will appear. Select the type of distance and the number of iterations from this menu. The vertical axis will display the chosen distances. See Section 2.6 for more details of the types of distances.

2.5 Plot Controls
All of the plot controls are inherited from the regression model prototype (see DG+RD). Some of the above described plots have additional plot controls and those are explained in the Plotting Methods section. I have added plot controls to the standard contour plot. These are described below.

2.5.1 Contour Plots
The default is for contour plots to have five plot controls.

- “Clear Values” is used to clear values which have been printed on the plot. In order to obtain the value of the function at a certain point, the user simply clicks at the desired spot. This is a handy method, but does often muddle the plot. If the plot becomes muddled, use the Clear Values button to clear up the plot.

- “Add Level(s)” is used to add a level or list of levels to the ones which are currently displayed.

- “Delete level(s)” is used to delete a level or levels from the levels currently displayed.

- “Double Grid” is used to ‘double’ the grid of points where the function has been evaluated. I put double in single quotes because if there are 6 lines on the horizontal axis part of the grid, Double Grid will increase that to 11 = 6 + 5. Each of the new lines will be half way between the old lines.

- “Add To Grid” is used to add lines to the grid of points where the function has been evaluated. For example, if the grid is 6 by 6 and I add one ‘point’ to the horizontal grid, the function will do 6 function evaluations in order to make the new grid 7 by 6.

2.6 Non-graphical Methods
Diagnostics in nonlinear regression also requires non-graphical output, including second derivative arrays, curvature calculations, etc. These method are summarized below.

- :first-derivative (&optional theta) Computes the two dimensional (n x p) first derivative matrix evaluated at theta. The elements of this matrix are V = (u_{ij}): 

\[ u_{ij} = \frac{\partial f}{\partial \theta_j} \big|_{\theta_i} \]

where f is the mean-function. If there is no theta argument, it will evaluate at the current parameter estimate of theta.

- :second-derivative (&optional theta) Computes the three dimensional (n x p x p) second derivative array evaluated at theta. The elements of this array are W = (w_{ijk}): 

\[ w_{ijk} = \frac{\partial f}{\partial \theta_j \partial \theta_k} \big|_{\theta_i} \]
where \( f \) is the mean-function. If there is no theta argument, it will evaluate at the current parameter estimate of theta.

- **exact-first-derivative** (f) Sets up the model to use the argument \( f \) as the first derivative function. The function \( f \) should be a function of \( \theta \) and \( x \) which evaluates to a list of the first partials of the mean function with respect to the parameters. In other words a list of \( \frac{\partial f}{\partial \theta_0}(\theta,x), \frac{\partial f}{\partial \theta_1}(\theta,x), \ldots, \frac{\partial f}{\partial \theta_{n-1}}(\theta,x) \). For example, the clarke data set has mean function

\[
g(\theta) = \theta_2 + \theta_1 e^{\theta x}
\]

Thus the first-derivative function should look something like this:

```lisp
(defun clarke-der #2'(lambda (th x)
    (list (* (second th) x (exp (* (first th) x)))
            (exp (* (first th) x))
            (repeat 1 (length x))))
)

(send clarke :exact-first-derivative clarke-der)
```

These commands will enable the clarke nonlinear regression object to use clarke-der as an exact first derivative function. If, for some reason the user wants to go back and use numerical derivatives, they should

```lisp
(send clarke :slot-value 'has-exact-first-derivative nil)
```

It is important to notice that this function is required to be vectorized. In other words, if \( x \) is a single element, it should return a list of the the derivatives. But, if \( x \) is a list, it should return a list of lists, and each of these lists should be the same length. This explains the (repeat 1 (length x)) in the definition of clarke-der.

- **exact-second-derivative** (h) Sets up the model to use the argument \( h \) as the second derivative function. The function \( h \) should be a function of \( \theta \) and \( x \) which evaluates to a list of the second partials of the mean function with respect to the parameters. The order is important and is as follows:

\[
\frac{\partial^2 f}{\partial \theta_0 \partial \theta_1}(\theta,x), \frac{\partial^2 f}{\partial \theta_0 \partial \theta_2}(\theta,x), \ldots, \frac{\partial^2 f}{\partial \theta_0 \partial \theta_{n-1}}(\theta,x), \frac{\partial^2 f}{\partial \theta_1 \partial \theta_0}(\theta,x), \frac{\partial^2 f}{\partial \theta_1 \partial \theta_1}(\theta,x), \frac{\partial^2 f}{\partial \theta_1 \partial \theta_2}(\theta,x), \ldots, \frac{\partial^2 f}{\partial \theta_{n-1} \partial \theta_{n-1}}(\theta,x)
\]

For example, to add and exact second derivative to the clarke data set discussed above, the following commands should be used:

```lisp
(defun clarke-sder
    #'(lambda (theta x)
        (let* ((etx (exp (* (first x) (first x)))
                (a (* (second th) (* (first x) 2) etx))
                (b (* (first x) etx))
                (n (length (first x)))
                (z (repeat 0 n)))
          (list a b z z z)))
)

(send clarke :exact-second-derivative clarke-sder)
```

These commands will enable the clarke nonlinear regression object to use clarke-sder as an exact second derivative function. If, for some reason, the user wants to go back and use numerical second derivatives, they should
Again as in the exact-first-derivative, it is important to notice that this function is required to be vectorized. In other words, if \( x \) is a single element, it should return a list of the the derivatives. But, if \( x \) is a list, it should returns a list of lists, and each of these lists should be the same length. This explains the (repeat 0 n) in the definition of clarke-sder.

- **:check-first-derivative** \( Q \) Evaluates the exact first derivative, \( A \), and the numerical first derivative, \( A \), arrays. It then returns the maximum \( \frac{E_{ij} - A_{ij}}{E_{ij}} \) along with the row \( i \) and the column \( j \) at which this maximum was attained. The user may send the model this message or simply select the Check First Derivative item in the menu.

- **check-second-derivative** \( Q \) Evaluates the exact second derivative, \( E \), and the numerical second derivative, \( A \), arrays. It then returns the maximum \( \frac{E_{ijk} - A_{ijk}}{E_{ijk}} \) along with the row \( i \), column \( j \) and depth \( k \) at which this maximum was attained. The user may send the model this message or simply select the Check Second Derivative item in the menu.

- **:u** \( Q \) Returns the \( Q \) matrix from the \( QR \)-decomposition of the first derivative matrix.

- **:r** \( Q \) Returns the \( R \) matrix from the \( QR \)-decomposition of the first derivative matrix.

- **:n-matrix** Returns an \( n \times n \) matrix. The columns of this matrix form an orthonormal basis for the orthogonal complement of the space spanned by the columns of the first derivative array.

- **:ap** \( Q \) Returns the \( p \times p \times p \) unscaled parameter effects curvature array, \( \Gamma^p \) as defined in Cook and Tsai (1990) or \( A^N \) as defined in Bates and Watts (1980).

\[
\Gamma^p = [U^T][R^{-T}WR]
\]

where \( U \), \( R \), and \( W \) are :u, :r, and :second-derivative respectively.

- **:ai** \( Q \) Returns the \( (n-p) \times p \times p \) unscaled intrinsic curvature array, \( \Gamma^I \) as defined in Cook and Tsai (1990) or \( A^T \) as defined in Bates and Watts (1980).

\[
\Gamma^I = [N^T][R^{-T}WR]
\]

where \( N \), \( R \), and \( W \) are :n-matrix, :r, and :second-derivative respectively.

- **:display-curvedatures** \( Q \) Displays the following curvatures:

  1. **:max-parameter-effects-curvature** \( Q \) \( (\&key \ (count-limit 20) \ (delta 1e-14)) \) Returns the maximum parameter effects curvature, \( \gamma^{T}_{\text{max}} \), as defined in Seber and Wild p. 143-4.

\[
\gamma^{T}_{\text{max}} = \max_{d \in \mathbb{R}^p} \rho \|d^T \Gamma^p d\|
\]

where \( \Gamma^p \) is :ap and \( \rho \) is \( \delta \sqrt{p} \). This maximum is calculated using the algorithm described in Seber and Wild, p. 150-4. First, define \( \gamma_d = \rho \|d^T \Gamma^p d\| \) for each \( d \), and find the standard basis \( e_i \) which maximizes \( \gamma_d \). We will use this \( e_i = d(0) \) as a starting value. The algorithm can then be summarized as follows:

(a) Using

\[
r(d) = \sum_{i=1}^{p} (d^T \Gamma^p_i d) \Gamma^p_i d
\]

calculate \( r(\alpha) = r(d(\alpha)) \) and \( r(\alpha) = r(\alpha) / \|r(\alpha)\| \).
(b) If \( p(\delta)^T d(\delta) < \delta \) then set \( d(\delta + 1) = r(\delta) \) and repeat step 1. Otherwise, return \( \gamma(\delta) \).

If the count-limit is reached without convergence, \( \gamma_{\text{count-limit}} \) is returned.

2. \texttt{:max-curvature- kp} \&\&\texttt{(key (count-limit 20) (delta .000001))} Returns \( K_P \) as defined in Cook and Tsai (1990 JASA), but without the \( \sqrt{c_\chi} \). If \( \sqrt{c_\chi} \) is desired, see \texttt{:curvature-c-alpha} below.

\[
K_P = \max_{||d||=1} \left| \left[ d^T \right] \left[ d^T \Gamma^P d \right] \right|
\]

where \( \Gamma^P \) is :ap above. This maximum is calculated in the following way. First, define \( \gamma(d) = [d^T][d^T \Gamma^P d] \), and find the standard basis \( e_i \) which maximizes \( \gamma(e_i) \). We will use this \( e_i = d(0) \) as a starting value. The algorithm can then be summarized as follows:

(a) Calculate \( \gamma(d(0)) \)
(b) Let

\[
d(\delta + 1) = \frac{d(\delta) \pm \frac{\gamma'(d(\delta))}{||d(\delta) - \gamma'(d(\delta))||}}{||d(\delta) \pm \frac{\gamma'(d(\delta))}{||d(\delta) - \gamma'(d(\delta))||}||}
\]

where the sign is the sign of \( \gamma(d(\delta)) \)
(c) Repeat step 2 until \( |\gamma(d(\delta)) - \gamma(d(\delta + 1))| < \delta \) or count-limit is exceeded.

3. \texttt{:max-curvature-ki} \&\&\texttt{()} Returns \( K_I \) as defined by Cook and Tsai (1990 JASA), but without the \( \sqrt{c_\chi} \). If \( \sqrt{c_\chi} \) is desired, see \texttt{:curvature-c-alpha} below.

\[
K_I = \max_{||d||=1} \left| \left[ d^T \right] \left[ d^T \Gamma^I N \right] \left[ \Gamma^I d \right] \right|
\]

where \( \Gamma^I \) is :ai above. This maximum is simply the maximum eigenvalue of the matrix \( \left[ d^T \right] \left[ d^T \Gamma^I N \right] \left[ d^T \right] \)

These curvatures help in evaluating some of the assumptions of a nonlinear model. Bates and Watts (1980) use the terms \textit{planar assumption} and \textit{uniform co-ordinate assumption}. A large intrinsic curvature \( (K_I) \) may indicate that the \textit{planar assumption} is unacceptable, and a large parameter effects curvature \( (K_P \text{ or } \gamma^2_{\text{max}}) \) may indicate that the \textit{uniform co-ordinate assumption} is not acceptable. Note that when the model is linear, all three of these curvatures are equal to 0.

To illustrate these curvatures, consider the data regarding turkey diets from Cook and Witmer. They calculate \( \gamma^2_{\text{max}} = 1.14 \) and \( K_P \sqrt{c_\chi} = 0.79 \). In order to test these numbers

\texttt{load "amino"

This will load the data and set up the graphics menu. Now select Curvatures from the Amino menu. This will print the following results:

Computing the maximum parameter effects curvature
Maximum Parameter Effects Curvature: 1.13933

Computing the Kp curvature
Maximum Kp curvature: 0.0136209

Computing the Ki curvature
Maximum Ki Curvature is 0.0834139

11
Next (send amino :curvature-c-alpha .05) to get $\sqrt{c_{\alpha}} = 57.9973$. This then gives

$$K_{p} \sqrt{c_{\alpha}} = 0.136309 \times 57.9973 = 7.89076$$

Notice that these numbers agree remarkably well considering that they are based on numerical second derivatives.

- :curvature-c-alpha (alpha) Returns $\sqrt{c_{\alpha}}$, a scale factor for the curvatures in Cook and Tsai (1990). $c_{\alpha} = 6^2 pF(p, n - p, \alpha)$
- :projected-residuals () Returns a list of the projected residuals as defined in Cook and Tsai (1985, Biometrika) formula (11).

$$P_{2} e = P_{1} e - P_{2/1} e$$

where $P_{1}$ is the projection matrix on to the orthogonal complement of the space spanned by the :first-derivative matrix. And, $P_{2/1}$ is the projection matrix on to the space spanned by the columns $P_{1} W_{ij}, i, j = 1, 2, \ldots, p$, where the $W_{ij}$ are the 'columns' of the :second-derivative array. This type of residual overcomes many of the shortcomings of the ordinary residuals, especially when the intrinsic curvature is large.

This function has been added to the :return-function list and will thus appear when either Plot Of or Scatterplot Of is selected from the menu.

- :projected-leverages () Returns a list of the projected leverages as defined in Cook and Tsai (1985, Biometrika). The projected leverages are the diagonal of the $P_{2}2$ matrix described above.

The Cook and Tsai paper uses the Ratkowsky data as an example. To reproduce plot (d) of Figure 1 in this paper, do the following:

(load "ratkowsky")
(def data #'(lambda ()
  (let* ((se (send rat :studentized-residuals))
         (pr (send rat :projected-residuals))
         (pl (send rat :projected-leverages))
         (sigma (sqrt (/ (sum (+ * 2)) (sum pl))))
         (sp12e (/ pr (* sigma (sqrt pl)))))
    (list (iseq 1 27) (< se sp12e))))

(send rat :make-plot data (list "x" "sd") :plot-controls t)

This will create a plot. If you click on the Join Points button and the Zero Line button, then plot (d) of Cook and Tsai (1988) will be reproduced. This is shown here in Figure 4

- :jacobian-leverages () Returns a list of the Jacobian leverages as defined by St. Laurent and Cook (1991 Biostatistics Technical Report) page 7 formula (9). $j_{i}$ is the ith diagonal element of the Jacobian leverage matrix, $J$.

$$J = V [V^{T} V - [e^{T} W]^{-1} e^{T} V^{T}$$

where $V$, $W$, and $e$ are the :first-derivative, :second-derivative and the :residuals respectively. The elements of $J$ are a measure of the rate of change in the predicted response as compared to the rate of change in the observed response. If $j_{i}$ is greater than 1, the $i$th case is said to exhibit superleverage since the effect of $y_{i}$ on $\hat{y}_{i}$ is magnified beyond the contribution of the observation itself.

This function has been added to the :return-function list and will thus appear when either Plot Of or Scatterplot Of is selected from the menu.

For example, to reproduce the round points in the second plot of Figure 7 in the paper by St. Laurent and Cook, do the following:
Figure 4: Scatterplot of sd = S(e) − S(\hat{P}_de) versus x where S(\cdot) denotes studentization

(load "swrat")

This will load the data from Seber and Wild page 478 with the Ratkowsky function. If you look at this file, you will note that it has included exact first and second derivative functions. This is because, if these functions are not included the Jacobian leverages will not match those in the paper. Finally, to produce the plot, select Plot of ... from the menu, scan down to find Case Numbers, Double click on this. Then scan to find Jacobian Leverages and double click on it. Then click on OK. A index plot of the Jacobian leverages will appear and is shown in Figure 5.

- :d-distances (key steps) Returns the D distances as defined by Ross (1987) formula (4.1).

\[ D_i = \frac{\phi_i^T V V \phi_i}{p \hat{\sigma}^2}, \quad i = 1, 2, \ldots, n \]

where \( \phi_i = \hat{\theta}_i - \hat{\theta}, \hat{\theta} \) is the steps steps estimate of \( \theta \) without using the \( i \)th case, and \( V \) is the first derivative matrix. If steps is not supplied, the program will ask for the number of steps. If steps is 1, the program will just return the cooks-distances. Since the calculation of these distances requires a nonlinear fit with out each case, and this is also required for the :fd-distances and :ls-distances, all of the calculations for those distances are done at the same time. This function along with the :fd-distances and :ls-distances functions are used by the :distance-plot method. It should be noted that because of the amount of time necessary to do the calculations, these values are stored in a slot. If the user has calculated the d-distances with 2 steps and has not changed the model, then

(send nonlin :d-distances :steps 2)

will returns the d-distance in a short period of time. But, if the model has changed, all of the distances will need to be recalculated.
Figure 5: Index plot of the $\hat{\gamma}_i$ for the swat model

- :fd-distances (key steps) Returns the $FD$ distances defined by Cook and Weisberg’s book formula (5.3.9).

$$FD_i = \frac{(f(\hat{\theta}) - f(\hat{\theta}_{(i)}))^T(f(\hat{\theta}) - f(\hat{\theta}_{(i)}))}{p \hat{\sigma}^2}$$

where $f$ is the mean function and again $\hat{\theta}_{(i)}$ is the steps steps estimate of $\theta$ without using the $i$th case. If steps is not supplied, the program will provide a dialog box for the number of steps.

- :ls-distances (key steps) Returns the LSD distances as defined in Ross (1987) formula (44)

$$LSD_i = n \ln \frac{S(\hat{\theta}_{(i)})}{S(\hat{\theta})}$$

where $S$ is the residuals sum of squares function and $\hat{\theta}_{(i)}$ is the steps steps estimate of $\theta$ without using the $i$th case. If steps is not supplied, the program will provide a dialog box for the number of steps.

3 Example

Bates and Watts use data for enzymes treated with Puromycin as an example. I will go through a few parts of this example in order to illustrate how to use the nonlin-model prototype. First

(l...pur"

This will load the data, the Michaelis-Menton model function $f(\theta, x) = \frac{\theta_1 x}{\theta_1 + x}$ and will display the least squares estimates for the parameters.
Figure 6: Plot of the reaction velocity versus substrate concentration for the Puromycin data along with the fitted curve \((\theta_0, \theta_1) = (212.684, 0.064121)\)

Least Squares Estimates:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter 0</td>
<td>212.684</td>
<td>(6.94589)</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>0.064121</td>
<td>(0.00827657)</td>
</tr>
</tbody>
</table>

R Squared: 0.961261  
Sigma hat: 10.9337  
Number of cases: 12  
Degrees of freedom: 10

In order to get a plot of the data, go to the PUR menu and select Plot Of ... This will call up a dialog box in which you should double click on Pred 0 and Response. Then click the OK button. This will then reproduce Figure 2.1 on page 34 of Bates and Watts. The user may want to see what to plot of the least squares estimator looks like on this plot. To do this you, must create a function of just x which uses the Michaelis-Menton function with \(\theta\) fixed at the estimate of theta. This can be done in the following way:

```lisp
(def th (send pur :theta-hat))  
(def fun #'(lambda (x) (funcall pur-fun th x)))  
(send (send pur :last-graph) :add-function fun 0 1.5)
```

This plot is shown here in Figure 6.

Moving along in Bates and Watts, Figure 2.12 on page 51 has plots of the studentized residuals versus the fitted values and a normal quantile plot of the studentized residuals. The residual plot can be easily reproduced by selecting the Stud. res. vs yhat option of the PUR menu. The normal quantile plot is produced by selecting Normal-plot from the PUR menu. These two plots are shown here in Figure 7.
The curvature measures described in Cook and Tsai can be obtained by selecting the Curvatures option in the PUR menu. This will then print the following:

**Computing the maximum parameter effects curvature**

Maximum Parameter Effects Curvature: 0.172263

**Computing the Kp curvature**

Maximum Kp curvature: 0.0109771

**Computing the Ki curvature**

Maximum Ki Curvature is 0.0962903

One might also want to investigate the influence of the points. To do this, select the Distances Plot option in the PUR menu. This will produce a plot of the Cook's Distances versus the case numbers. Notice that case 0 has a Cook's distance of approximately 0.44 while all of the other cases have Cook's distances of 0.1 or less. This may lead you to investigate what happens when case 0 is deleted. This can be done in the following way. In the Distance Plot, use the mouse to select the point corresponding to case 0. Then select Delete Selection from the plot menu. This will cause the estimates to be recomputed and the new Cook's distances to be plotted. This will give Figure 8.

There is much more that can be easily done to analyze this data set. The reader is encouraged to use some of the other methods described in this paper.

## 4 How it Works

All of the methods described above interact and inherit from the regression-model-proto described in DG+RD. However, because of computational concerns, much of the information is not calculated until it is needed. For example, if none of the distance measures is plotted, and a case is deleted, the parameter estimates are recomputed, but the distance measures are not. A flag is set however, so that, when a distance measure is needed, they will be recomputed with the proper case deletion. The same thing is true for any plots that depend on the second derivative array.

The code is organized into several files named `nonlin.lsp`, `nonlin-regr1.lsp`, `nonlin-regr2.lsp`, `nonlin-regr3.lsp`, `nonlin-regr4.lsp`, and `nonlin-regr5.lsp`. The new contour plot methods are in `new-contour-plot.lsp`. And some miscellaneous functions are found in `fun1.lsp`. 

16
5 Summary

The methods described in this paper are some of the useful methods for analyzing nonlinear regression models. The XLISP-STAT environment allows many more methods that you may want to employ. If this is the case, add those methods to your installation.

6 References


