CP-628 15 October 1997

A Practical Guide to Least-Squares Fitting

R.J. Le Roy

Guelph-Waterloo Centre for Graduate Work in Chemistry
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
Electronic mail address: leroy@uwaterloo.ca

University of Waterloo

Chemical Physics Research Report

A Practical Guide to Least-Squares Fitting

Robert J. Le Roy

Guelph-Waterloo Centre for Graduate Work in Chemistry
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

A. Non-Weighted Linear Least Squares Fits

A least squares fit seeks to find the values of a set of M parameters p_j (for $j = 1 - M$) which will optimize the agreement between a set of N experimental data $y_o(i)$ (for $i = 1 - N$) and calculated values for these data $y_c(\{p_j\}; i)$ ($i = 1 - N$) generated from a model defined by the M parameters. A *linear* least squares problem is one in which the calculated function values defined by the model are *linearly* dependent on the parameters of interest. When this is true, the function defining the model can be written

$$y_c(i) \;\equiv\; y_c(\{p_j\}; i) \;=\; \sum_{j=1}^{M} p_j\, \phi_j(i) \tag{1}$$

where $\phi_j(i)$ ($j = 1 - M$) are known functions which do not depend on the values of the parameters $\{p_j\}$. Another way of stating this definition is to say that that a least squares problem is *linear* if the partial derivatives

$$\phi_j(i) \;\equiv\; \left(\frac{\partial y_c(\{p_j\}; i)}{\partial p_j}\right)_{\{p_k\},\, k \neq j} \tag{2}$$

have absolutely no dependence on the p_j's.

A simple spectroscopic example of a linear least squares problem is that of fitting the frequencies of the R-lines of a diatomic molecule microwave spectrum to the expression

$$y_c(i) \;=\; \nu_c(i) \;=\; F(J_i+1) - F(J_i) \;=\; B\,2(J_i+1) \;-\; D\,4(J_i+1)^3 \tag{3}$$

Here the data are labeled by the values of the rotational quantum number $J = J_i$, the two parameters are the "inertial" rotational constant $B = p_1$ and the leading centrifugal distortion constant $D = p_2$, and the partial derivatives of the predicted transition frequencies $\nu_c(i)$ with respect to these parameters,

$$\phi_1(i) \;=\; \phi_1(J_i) \;=\; 2(J_i+1) \qquad \text{and} \qquad \phi_2(i) \;=\; \phi_2(J_i) \;=\; 4(J_i+1)^3 \tag{4}$$

have no dependence on the values of the parameters of this model.

The above sample problem becomes *non*-linear if one modifies the model by introducing the Kratzer relation for the centrifugal distortion constant, $D = 4B^3/\omega^2$. In this case the model becomes

$$\nu_c(i) \;=\; B\,2(J_i+1) \;-\; [\,16\,B^3/\omega^2\,]\,(J_i+1)^3 \tag{5}$$

and the parameters are now the inertial rotational constant B and the vibrational frequency ω. This expression is *not* linearly dependent on B or ω, since the derivatives

$$\frac{\partial \nu_c(i)}{\partial B} \;=\; \phi_1(i) \;=\; 2(J_i+1) \;-\; 48\,(B/\omega)^2\,(J_i+1)^3 \tag{6}$$

and

$$\frac{\partial \nu_c(i)}{\partial \omega} \;=\; \phi_2(i) \;=\; 32\,(B/\omega)^3\,(J_i+1)^3 \tag{7}$$

do depend on the values of these parameters. Of course this is a trivial type of non-linear problem, in that a unique solution for it may be determined by manipulating the parameters of the related linear problem of Eq. (3).

However, it serves to illustrate the meaning of "non-linearity" in this context. In this section, however, our attention will be restricted to *linear* least squares models such as that of Eq. (3).

If all of the data are weighted equally, the least squares fitting problem is concerned with determining the set of parameter values for which the sum of squares of deviations,

$$SSD \;\equiv\; SSD(\{p_j\}) \;=\; \sum_{i=1}^{N} \Big[y_o(i) - y_c(\{p_j\};i) \Big]^2 \;=\; \sum_{i=1}^{N} \Big[y_o(i) - \sum_{j=1}^{M} p_j\, \phi_j(i) \Big]^2 \quad , \tag{8}$$

is a minimum. As discussed in standard texts and references,[1,2] this involves requiring the partial derivatives of SSD with respect to each of the model's parameters p_k to simultaneously equal zero:

$$\frac{\partial\, SSD}{\partial p_k} \;=\; 0 \;=\; -2 \sum_{i=1}^{N} \Big[y_o(i) - y_c(\{p_j\};i) \Big] \frac{\partial y_c(\{p_j\};i)}{\partial p_k} \;=\; -2 \sum_{i=1}^{N} \Big[y_o(i) - \sum_{j=1}^{M} p_j\, \phi_j(i) \Big]\, \phi_k(i) \tag{9}$$

Rearranging these expressions yields a set of M linear equations (one for each of $k=1$ to M) in the M unknown parameters $\{p_j\}$:

$$\sum_{i=1}^{N} [\, y_o(i)\, \phi_k(i)\,] \;=\; \sum_{j=1}^{M} p_j \left[\sum_{i=1}^{N} [\, \phi_j(i)\, \phi_k(i)\,] \right] \tag{10}$$

Solving the above set of linear equations is of course a simple matter of linear algebra, and devising a highly stable computational algorithm for doing so is sensibly left to numerical computation experts. However, when such an algorithm is properly packaged, a user may exploit it to solve *any* linear least squares problem of interest. The only input information required is the experimental data and a knowledge of the partial derivatives of each calculated data point $y_c(\{p_j\};i)$ with respect to each parameter of the model, p_k. If properly set up, the output will provide (in addition to the desired parameter values $\{p_j\}$) a range of statistical information regarding the parameters and the fit (see §E). The Fortran subroutines LLSQF and LLSQFVL described below (and partially listed in the Appendix) have been designed precisely for this purpose, and may be obtained from the author on request. While a variety of equivalent packages may be available to a user, these routines are particularly robust and stable, and it is convenient to use their capabilities to illustrate the discussion of other features of least squares analyses. Thus, we precede our discussion of non-linear fits and other matters by a short outline of the characteristics of LLSQF and LLSQFVL .

The presentation to this point gives us the ability to use a packaged routine such as LLSQF to perform a least squares fit of experimental data to *any* function expressed in the linear form of Eq. (1). While we have explicitly considered only the case of fits to equally-weighted data, the generalization to the case of non-equal data uncertainties is quite straightforward. However, since those considerations are the same for both linear and non-linear fits, their discussion is postponed to §D.

B. Capabilities, Use and Source of the Routines LLSQF & LLSQFVL

For a least squares fit to *any* linear model written in the form of Eq. (1), a user of LLSQF must write a calling program which will read in the experimental data and prepare the partial derivative array

$$\mathrm{DYDP}(i,k) \;=\; \left[\frac{\partial y_c(\{p_j\};i)}{\partial p_k} \right]_{\{p_l\},\, l\neq k} \;=\; \phi_k(i) \tag{11}$$

which is dimensioned in the calling program as $\mathrm{DYDP}(\mathrm{MXDATA},\mathrm{MXPARM})$. The only other information required as input is the set of integers specifying the number of data $\mathrm{NDATA}=N$ and the number of parameters $\mathrm{NPARM}=M$, a linear array containing the experimental data $\mathrm{YO}(i)$, and a linear array $\mathrm{YU}(i)$ containing the uncertainties in each of these experimental observables. If all of the data are weighted equally, the values of

YU(i) should all be set equal to unity; for other cases, see the discussion in §D below.

For a several-parameter fit to very large numbers of data (say $N \gtrsim 10^3 - 10^6$), the requisite partial derivative array DYDP may be sufficiently large that the associated computer memory requirements make the computation unwieldy. For this case an alternate version of this routine, called LLSQFVL , has been developed. It's argument list is the same as that of LLSQF except that array DYDP is omitted and the requisite partial derivatives are calculated for one datum at at time by a user-prepared subroutine called DYIDPJ (see Appendix).

On return from a call to LLSQF (or LLSQFVL), in addition to the recommended values of the least squares fitted parameters, $\mathrm{PV}(j) = p_j$ $(j = 1 - M)$, one is provided with the correlated 95% confidence limit uncertainty in each parameter $\mathrm{PU}(j)$ (see §E), with the "truncation tolerance" or precision to which each parameter should be quoted $\mathrm{PS}(j)$ (see §F), with the standard error of the fit, $\mathrm{SERR} = \sqrt{SSD/(N-M)}$, and with the "correlation matrix" **CM** which is obtained on normalizing the variance-covariance matrix. This last quantity contains information regarding the degree of interparameter correlation, and may be used to compute the variance or uncertainty in *any* quantity calculated from the parameters determined by the fit (see §E below).

A copy of LLSQF will be returned to anyone who sends a request for it by electronic mail to: *leroy@theochem.uwaterloo.ca* . Alternately, persons with an account on an INTERNET-connected computer may pick up copies for themselves using standard anonymous *ftp* protocols. In the latter case, after issuing the command *ftp theochem.uwaterloo.ca* , the response to the userid prompt should be *anonymous* and the response to the password prompt should be the person's e-mail address. Once a successful connection is made, the desired files may be copied from subdirectory *pub/leroy* . For the record, I would appreciate being notified (by e-mail to *leroy@theochem.uwaterloo.ca*) about any files/programs picked up in this way.

C. Non-Linear Least Squares Fits

A linear least squares fit has exactly one solution, consisting of a single unique set of $\{\mathrm{PV}(j)\}$ values and their uncertainties $\{\mathrm{PU}(j)\}$. For a non-linear problem, however, there may in general be a number of sets of $\mathrm{PV}(j)$'s for which the SSD function has local minima, and there exists no *a priori* way of knowing which of them defines the absolute minimum. When considering the results of *any* non-linear least squares fit, this fact must be kept in mind. In practise, however, many non-linear fitting problems appear to have only one solution, and they are often at least "locally" linear, so that converged fits may be obtained by performing an iterative series of linear fits until convergence is achieved. While more sophisticated iteration schemes have been developed,[2] only the simple "locally linear" approach is discussed below, as it most clearly illustrates the essential methodology, and is the easiest to program.

The defining property of non-linear least squares fits is the fact that the partial derivatives of Eq. (2) *do* depend on the values of the parameters, so no analog of the set of linear equations of Eq. (10) may be set up without having some initial trial estimates of the p_j's to use in calculating the partial derivatives $\{\phi_j(i)\}$. However, if plausible initial trial parameter values are somehow obtained, one may proceed in the following manner. Since (linear or not) the values of $y_c(\{p_j\};i)$ calculated from the model have some explicit dependence on the parameter values $\{p_j\}$, one may always expand y_c as a multidimensional Taylor series about those initial trial values. In particular, we may express the values of $y_c(\{p_j\};i)$ associated with the optimum (but not yet known!) parameter values $\{p_j(opt)\}$ in terms of a Taylor series expansion about our initial trial values $\{p_j(trial)\}$:

$$y_c \;=\; y_c(\{p_j(trial)\};i) \;+\; \sum_{j=1}^{M} \frac{\partial y_c(i)}{\partial p_j}\,\Delta p_j \;+\; \frac{1}{2}\sum_{j=1}^{M}\sum_{l=1}^{M} \frac{\partial^2 y_c(i)}{\partial p_j\,\partial p_1}\,\Delta p_j\,\Delta p_l \;+\; \cdots \qquad (12)$$

where $\Delta p_j = p_j(opt) - p_j(trial)$ are the corrections to the initial trial parameter values, which we wish to determine.

Substituting Eq. (12) into the definition of SSD and retaining only terms linear in the parameter corrections yields the equation:

$$SSD \;=\; SSD(\{\Delta p_j\}) \;=\; \sum_i \left\{ [\, y_o(i) - y_c(\{p_j(trial)\},\, i)\,] \;-\; \sum_{j=1}^{M} \Delta p_j\; \phi_j(i) \right\}^2 \tag{13}$$

This expression has exactly the same form as the last version of Eq. (8), except that $y_o(i)$ is replaced by $[\, y_o(i) - y_c(\{p_j(trial)\},\, i)\,]$ and p_j is replaced by Δp_j. By analogy with the linear least squares problem, we wish to find values of the Δp_j's for which the partial derivatives of SSD with respect to further parameter changes are all simultaneously zero. This requirement leads to a set of M linear equations (for $k = 1$ to M) in the M unknowns Δp_j:

$$\sum_{i=1}^{N} [\, y_o(i) - y_c(\{p_j(trial)\},\, i)\,]\, \phi_k(i) \;=\; \sum_{j=1}^{M} \Delta p_j \left\{ \sum_{i=1}^{N} [\, \phi_j(i)\, \phi_k(i)\,] \right\} \tag{14}$$

As in the linear least squares problem, this set of equations may be solved using a standard package such as LLSQF to yield estimates of the *changes* Δp_j in the parameter values required to optimize the model. In this case, however, the input array $\mathrm{YO}(i)$ consists of the differences $[\, y_o(i) - y_c(\{p_j(trial)\},\, i)\,]$ defined by the current trial parameter values, rather than the experimental observables $y_o(i)$ themselves, and the quantities $\mathrm{P}(j)$ returned by the subroutine are the incremental parameter changes Δp_j rather than the parameters themselves. At the same time, the parameter uncertainties $\mathrm{PU}(j)$ and truncation tolerances $\mathrm{PS}(j)$ retain their usual significance, since the uncertainty and precision associated with a parameter change from some known starting value is equivalent to the uncertainty in and required precision of the parameter value itself.

Because of the approximations associated with neglect of the higher-order terms in Eq. (12) (i.e., because the partial derivatives $\phi_j(i)$ change with the parameter values), the changes $\{\Delta p_j\}$ yielded by a single application of this procedure will only be approximate, so it is necessary to repeat it iteratively until these changes converge to zero. The overall procedure then consists of the following steps:

(i) Read in the data and generate some initial set of trial values for the parameters defining the model.

(ii) Prepare the arrays $\mathrm{YO}(i) = [\, y_o(i) - y_c(\{p_j(trial)\},\, i)\,]$ and $\mathrm{DYDP}(i,j) = \phi_j(\{p_j\};\, i)$, and call the linear least squares routine LLSQF .

(iii) Update the parameter values: $p_j(new) = p_j(trial) + \Delta p_j$, and test for convergence. If the changes are "negligible" (see below), stop; otherwise return to step (ii).

A Simple Example

Consider the problem of determining the values of parameters *a, b* and *c* by fitting a set of equally-weighted data to the model function

$$y_c(\{p_j\};\, i) \;=\; y_c(a,b,c\,;\, x_i) \;=\; a \;+\; b\, e^{-c\, x_i} \tag{16}$$

For any given set of trial values of the parameters a , b and c , one can readily determine the partial derivatives:

$$\mathrm{DYDP}(i,\,1) \;=\; \phi_1(i) \;=\; \partial y_c(i)/\partial a \;=\; 1 \tag{17}$$

$$\mathrm{DYDP}(i,\,2) \;=\; \phi_2(i) \;=\; \partial y_c(i)/\partial b \;=\; e^{-c\, x_i} \tag{18}$$

$$\mathrm{DYDP}(i,\,3) \;=\; \phi_3(i) \;=\; \partial y_c(i)/\partial c \;=\; -\, x_i\, b\, e^{-c\, x_i} \tag{19}$$

The fact that (two of) these expressions contain values of the parameters underlines the fact that fitting data to Eq. (16) is a non-linear least squares problem. Moreover, *un*like the model of Eq. (5), it cannot be made equivalent to some linear model.

To fit a set of data to Eq. (16), it is necessary to first determine a set of trial parameters to use in generating the partial derivatives of Eqs. (17)-(19). When this is done and the DYDP(i,j) array prepared, a call to LLSQF would then yield proposed changes to the three parameters: Δa, Δb and Δc. These quantities may then be added to the original trial parameters, and if the chosen convergence criterion is not satisfied, the fit continued by using these updated parameters in step (ii) of the procedure outlined above.

Obtaining Initial Trial Parameter Values

It important to realize that there is no general way of determining a "reasonable" set of trial parameter values with which to initialize an iterative non-linear least squares fit. For each model and data set, a user must utilize their own physical and/or mathematical intuition, while making use of any physical constraints which may be known for the problem. In some cases, values of some parameters in the model may be fixed at zero (or some other values) while a preliminary series of iterations optimizes initial estimates of others, and then the full fit is allowed to proceed with all parameters set free. However, this is an area in which the best guide is experience.

Testing for Convergence

One can readily think of a number of criteria which might be used to test for convergence of the iterative procedure described above. For example, one may require that the absolute $|\Delta p_j|$ or relative $|\Delta p_j/p_j|$ parameter changes be smaller than some specified criterion, or that the relative changes in the standard error of the fit (SERR) from one iteration to the next be less than some chosen tolerance. Another possibility is to require that the magnitude of the change in each parameter $|\Delta p_j|$ be less than some specified fraction of the parameter uncertainty PU(j). However, these criteria overlook the essential requirement for the utility of parameter values determined from a fit. It is that: *quantities calculated from the converged parameters should agree with the input data to within a small fraction of the uncertainty of the fit*. With this criterion in mind, a more selective (and usually more stringent) criterion may readily be defined.

The "sensitivity" of a particular datum to a given parameter value is defined by the partial derivative of Eq. (2). For small Δp_j, the change in $y_c(i)$ caused by this parameter change is $\delta y_c(i,j) \approx \phi_j(i)\,\Delta p_j$. A reasonable convergence criterion would therefore be to require that for each parameter j, the root mean square value of $\delta y_c(i,j)$ be smaller than some chosen fraction f of the overall standard error of the fit. With (following Watson,[3] see §F) $f \equiv 0.1/M$, this condition may be restated as the requirement that for each parameter p_j, the magnitude of the change Δp_j be smaller than the predicted "parameter sensitivity" (as returned by LLSQF and LLSQFVL):

$$\mathrm{PS}(j) \;=\; \frac{0.1}{M}\,SERR \Big/ \left\{ \sum_{i=1}^{N} \phi_j(i)^2 / N \right\}^{1/2} \qquad (15)$$

This is the test which I recommend be used in step (iii) above.

One apparent difficulty with this convergence criterion may arise in highly-correlated or many-parameter fits where numerical noise due to finite computer precision may prevent this limit from actually being reached. However, one flag for such behaviour would be an increase in $\max_j |\Delta p_j/\mathrm{PU}(j)|$ from one cycle to the next. If this occurs and (say) $\max_j |\Delta p_j/\mathrm{PU}(j)| < 0.1$, physical convergence has been achieved and the iteration procedure can simply be stopped.

D. Weighting in Linear and Non-Linear Least Squares Fits

If all of the data used in a fit are weighted equally and the data uncertainties YU(i) input to subroutine LLSQF set equal to unity, the standard error of the fit SERR has the same units as the data values, and may readily be related to the precision of those observables. However, if the data set being fitted includes different *types* of physical quantities (e.g., spectroscopic line positions and intensities), one must approach the problem differently. In a formal treatment, this would involve introduction of a diagonal weight matrix whose non-zero elements are the inverse squares of the uncertainties in the data, $w(i) = 1/\mathrm{YU}(i)^2$, and minimization of *SSD* of Eq. (8) would be replaced by minimization of the χ^2 function:

$$\chi^2 \;=\; \chi^2(\{p_j\}) \;=\; \sum_{i=1}^{N} w(i)\left[y_o(i) - y_c(\{p_j\};i)\right]^2 \;=\; \sum_{i=1}^{N}\left[\,[\,y_o(i) - y_c(\{p_j\};i)\,]/\mathrm{YU}(i)\right]^2 \tag{20}$$

In practise, however, all of the changes associated with the use of non-unit uncertainties are accounted for internally inside LLSQF and LLSQFVL if the user causes the array $\mathrm{YU}(i)$ to contain the uncertainties in the individual experimental data. In this case, the quantity SERR returned from LLSQF is actually the dimensionless quantity $\sqrt{\chi^2/(N-M)}$, and in the calculation of the quantities $\mathrm{PS}(j)$ inside LLSQF (see Eq. (15)), the actual partial derivatives $\phi_j(i)$ are replaced by the scaled quantities $\phi_j(i)/\mathrm{YU}(i)$. Thus, while the matrix formulation of the least squares problem with non-equal weights may appear complicated,[1] the practical treatment of such problems is quite straightforward.

E. The Correlation Matrix and Uncertainties the Parameters and in Calculated Properties

The partial derivative array $\mathrm{DYDP}(i,j)$ of Eq. (11) has N rows, one for each experimental datum, and M columns, one for each parameter. The variance-covariance matrix **V** is then defined (for the case of unit weights $\mathrm{YU}(i) = 1$) as the inverse of the matrix obtained on multiplying **DYDP** by its transpose:

$$\mathbf{V} \;=\; \left[\mathbf{DYDP}^t \cdot \mathbf{DYDP}\right]^{-1} \tag{21}$$

For the more general case of non-equal weights, **V** is defined by the fact that the elements of its inverse $\mathbf{V}^{-1}$ are weighted sums of products of the individual partial derivatives:

$$(\mathbf{V}^{-1})_{j,k} \;=\; \sum_{i=1}^{N} w(i)\,\phi_j(i)\,\phi_k(i) \tag{22}$$

The "correlation matrix" **CM** returned by LLSQF is then defined as the matrix obtained on "normalizing" the variance-covariance matrix by dividing each row and column by the square root of the associated diagonal elements:

$$(\mathbf{CM})_{i,j} \;=\; \mathrm{V}_{i,j}\big/\sqrt{\mathrm{V}_{i,i}\,\mathrm{V}_{j,j}} \tag{23}$$

Of course, all of this manipulation is automatically done inside LLSQF (or its equivalent), so that all a user need consider is the meaning and use of **CM**.

The uncertainty associated with a parameter value yielded by a least squares fit has two sources. The first is the direct "sensitivity" of the data to this parameter, information contained in the partial derivatives $\phi_j(i)$. This is a measure of how much *SSD* or χ^2 would increase if that parameter was changed while all others were held fixed. The second source of uncertainty is "interparameter correlation". It reflects the fact that if the effect of a given change in (say) parameter-*k* on the calculated properties $\{y_c(\{p_j\};i)\}$ (and hence on *SSD*) may be partially compensated for by a correlated change in one or more of the other parameter(s), the overall range of uncertainty associated with parameter-*k* could be much larger than that due only to its sensitivity. Of course the parameter uncertainties $\mathrm{PU}(k)$ returned by LLSQF take account of both these sources of error. They are calculated as

$$\mathrm{PU}(k) \;=\; f_{95}(N-M)\,SERR\,\sqrt{\mathrm{V}_{k,k}} \tag{24}$$

where $f_{95}(N-M)$ is the 95% student t-value for a fit with $N-M$ degrees of freedom.

One use of the correlation matrix is in indicating which parameters are highly correlated with one another. In general, diagonal elements of **CM** are precisely equal to unity while the off-diagonal elements $(\mathbf{CM})_{j,k}$ have magnitudes (< 1) which indicate the degree to which the effect on *SSD* of changes in a particular p_j can be compensated for by changes in p_k (and *vise versa*). If $|(\mathbf{CM})_{j,k}|$ is close to unity (say ≥ 0.95), it means that parameters j and k are "highly correlated", and that much of their uncertainty is due to this correlation. An example of a real correlation matrix, obtained from a combined-isotope fit of infrared data for the Van der Waals molecules H_2- and D_2-Kr to a model potential energy surface, is shown below [from *J. Chem. Phys.* **86**, 837 (1987)].

parameter	ε^{00}	R_e^{00}	β	ε^{01}	R_e^{01}	ε^{20}	R_e^{20}	ε^{21}	C_8^{40}
ε^{00}	1.00								
R_e^{00}	-0.91	1.00							
β	-0.98	0.97	1.00						
ε^{01}	0.68	-0.67	-0.72	1.00					
R_e^{01}	-0.95	0.94	0.96	-0.51	1.00				
ε^{20}	0.24	-0.24	-0.24	-0.23	-0.38	1.00			
R_e^{20}	-0.66	0.62	0.66	-0.78	0.52	0.56	1.00		
ε^{21}	-0.40	0.40	0.39	-0.18	0.36	0.06	0.34	1.00	
C_8^{40}	0.01	0.01	0.01	-0.09	-0.03	0.20	0.16	-0.08	1.00

The main quantitative use of the correlation matrix is in generating the proper correlated statistical uncertainty in any quantity calculated from the parameters yielded by a fit. In particular, if a property $F(\{p_j\})$ may be calculated from the fitted parameters $\{p_j\}$, its squared 95% confidence limit uncertainty may be calculated from the expression

$$u(F)^2 \;=\; \mathbf{PD}^t \cdot \mathbf{CM} \cdot \mathbf{PD} \qquad (25)$$

where **PD** is an $M \times 1$ column matrix of elements $\mathrm{PD}(j) = \mathrm{PU}(j)\,[\,\partial F/\partial p_j\,]$, for $j = 1$ to M. Thus, simply combining the results of the fit with the partial derivatives of the property of interest with respect to the fit parameters yields proper estimates of the uncertainty in the predicted property. For example, when dealing with spectroscopic data, this would allow one to calculate the real uncertainties in line positions for very high J's, beyond the range of the input data, predicted using molecular constants determined from a fit to the model of Eq. (3).

F. Rounding of Parameters Determined from Least Squares Fits

For the sake of simplicity, one always wishes to minimize the number of significant digits to which each parameter determined from a fit must be reported. In deciding how to effect this rounding, it is tempting to use the values of the parameter uncertainties $\{\mathrm{PU}(j)\}$ as a guide, and to round each parameter off at, say, 10% or 1% of this value. However, this neglects the fact that most of this uncertainty is usually due to interparameter correlation, and hence that even such *statistically* insignificant changes as (say) $0.01 \times \mathrm{PU}(j)$ may substantially affect the predictions yielded by the model.[3] In contrast, the direct "sensitivity" of the data to each parameter provides a much more useful measure of the precision to which it must be reported.

This problem has in fact already been addressed in the context of of our discussion of how to best test for convergence of an iterative non-linear least squares fit. The definition of the "parameter sensitivities" $\{\mathrm{PS}(j)\}$ of Eq. (15) shows that these quantities provide reasonable bounds on the truncation errors which may be tolerated when rounding off the fitted parameters. Thus, following the recommendation of Watson,[3] it is proposed that the $\mathrm{PS}(j)$ values of Eq. (15) be used to define the precision to which parameters obtained from both linear and nonlinear fits should be reported, as well as in testing for convergence of the latter.

G. Determining Derivatives-By-Differences for Use in Least-Squares Fits

As discussed above, performing linear or non-linear least squares fits requires a knowledge of the partial derivatives of the predicted values of the observables with respect to each of the parameters of the model. It is always best to generate these partial derivatives analytically or from closed form expressions. However, sometimes this seems impossible or very inconvenient, and it is necessary to generate the required derivatives from a numerical first-difference expression:

$$\frac{\partial f}{\partial p} \approx \frac{f\,(p+\Delta p)-f\,(p-\Delta p)}{2\cdot\Delta p} \tag{26}$$

In this case, two competing considerations must be taken into account when choosing the parameter stepsize Δp .

(i) Numerical analysis tells us that the smaller this increment, the better the estimate of the derivative. However, no further improvement is achieved when the step becomes so small that the error term is smaller than the computer precision limit.

(ii) While consideration-(i) favours ever smaller parameter increments, finite computer precision means that if Δp becomes too small the difference in the numerator on the right hand side of Eq. (26) loses all of its significant digits, and the resulting derivative becomes meaningless.

We wish to delineate how these two considerations may be optimally reconciled.

With regard to consideration-(i), a Taylor series expansion for the function $f\,(p)$ yields:

$$f\,(p+\Delta p) = f\,(p) + \frac{\partial f}{\partial p}\Delta p + \frac{\partial^2 f}{\partial p^2}\frac{(\Delta p)^2}{2!} + \frac{\partial^3 f}{\partial p^3}\frac{(\Delta p)^3}{3!} + \cdots \tag{27}$$

$$f\,(p-\Delta p) = f\,(p) - \frac{\partial f}{\partial p}\Delta p + \frac{\partial^2 f}{\partial p^2}\frac{(\Delta p)^2}{2!} - \frac{\partial^3 f}{\partial p^3}\frac{(\Delta p)^3}{3!} + \cdots \tag{28}$$

As a result, the estimate of the partial derivative provided by Eq. (26) becomes

$$\frac{f\,(p+\Delta p)-f\,(p-\Delta p)}{2\cdot\Delta p} = \frac{\partial f}{\partial p} + \frac{\partial^3 f}{\partial p^3}\frac{(\Delta p)^2}{3!} + \cdots = \frac{\partial f}{\partial p}\left[1 + \left(\frac{\partial^3 f/\partial p^3}{\partial f/\partial p}\right)\frac{(\Delta p)^2}{3!} + \cdots\right] \tag{29}$$

In practical numerical work, the first difference approximation for this derivative becomes essentially exact when the second term in square brackets on the right hand side of Eq. (29) becomes smaller than the machine precision constant ε (typically $\varepsilon \approx 10^{-16}$ for REAL*8 arithmetic on common computers). In other words, obtaining optimal accuracy using the first difference approximation of Eq. (26) requires that

$$\Delta p < (6\,\varepsilon)^{1/2} \left|\frac{\partial f/\partial p}{\partial^3 f/\partial p^3}\right|^{1/2} \tag{30}$$

With regard to consideration-(ii), we wish to avoid an excessive loss of significant digits on taking the difference in the numerator of Eq. (26). In other words, we wish to ensure that this difference of approximately $\{2\cdot\Delta p\cdot(\partial f/\partial p)\}$ is a significant fraction of the magnitude of the function value f itself. In particular, requiring that no more than half of the machine precision significant digits be lost on taking this difference means that:

$$\Delta p > \varepsilon^{1/2}\,\left|\,f\,(p)/2\,(\partial f/\partial p)\right| \tag{31}$$

In attempting to reconcile Eqs. (30) and (31), we note that for some types of algebraic functions,

$$\left|\frac{\partial f/\partial p}{\partial^3 f/\partial p^3}\right|^{1/2} \approx \left|\frac{\partial f/\partial p}{\partial^2 f/\partial p^2}\right| \approx \left|\,f\,(p)/(\partial f/\partial p)\right| \tag{32}$$

As a result, an optimal ballance between considerations (i) and (ii) will often be achieved by setting

$$\Delta p \;\approx\; \varepsilon^{\frac{1}{2}} \left| f(p) \big/ (\partial f/\partial p) \right| \qquad or \qquad \varepsilon^{\frac{1}{2}} \left| \frac{\partial f/\partial p}{\partial^2 f/\partial p^2} \right| \tag{33}$$

In the context of a least squares fit, one would usually wish to replace the values of $f(p)$ and its derivatives in these expressions by values averaged over the data set.

References

[1] See, e.g., D. Albritton, A.L. Schmeltekoff and R.N. Zare, Chapter 1 (pp. 1-67) of *Molecular Spectroscopy: Modern Research*, Volume 2 (K.N. Rao editor, Academic Press, New York, 1976).

[2] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes. The Art of Scientific Computing* (Cambridge University Press, Cambridge, UK, 1986).

[3] J.K.G. Watson, *J. Mol. Spectrosc.* **66**, 500 (1977).

APPENDIX: Calling Sequence and Partial Listings for LLSQF & LLSQFVL

```
c************************************************************************
      SUBROUTINE LLSQF(NDATA,NPARM,MXDATA,MXPARM,YO,YU,DYDP,DY,P,PU,PS,
     1                 CM,SERR)
c**  Program for performing linear least squares fits using orthogonal
c  decomposition of the Design (partial derivative) matrix.
c** This version of the program is designed for the data sets of modest
c  size where it is convenient to generate and store the complete
c  partial derivative matrix prior to calling LLSQF.  If this is not the
c  case, subroutine version LLSQFVL, which generates this partial
c  derivative array one row at a time through calls to a user-supplied
c  subroutine, should be used.
c
c** On entry: NDATA  is the number of data to be fitted (.le.MXDATA)
c             NPARM  the number of parameters to be varied (.le.MXPARM)
c                 If NPARM.le.0 , simply calculate the (weighted RMS
c                 deviation)=SERR  from the input YO(i) & YU(i)’s
c             MXDATA & MXPARM are array dimension parameters (see below)
c                 Internal array sizes currently assume  MXPARM .le. 60
c             YO(i)  are the NDATA ’observed’ data;  for iterative
c                  non-linear fits these are:  [Y(obs,i) - Y(trial,i)]
c             YU(i)  are the uncertainties in these YO(i) values
c             DYDP(i,j)  is the partial derivative array  dYO(i)/dPV(j)
c
c** On Exit: PV(j)  are the fitted parameter values;  for iterative
c                  non-linear fits these are the parameter changes
c            PU(j) are 95% confidence limit uncertainties in the PV(j)’s
c            PS(j) are ’parameter sensitivities’ for the PV(j)’s, defined
c               such that the RMS displacement of predicted data  due to
c               rounding off parameter-j by PS(j) is .le. SERR/10*NPARM
c            SERR  is the standard error of the fit
c            DY(i) is the array of differences  [YO(i) - Ycalc(i)]
c            CM(j,k)  is the correlation matrix obtained by normalizing
c   variance/covariance matrix:  CM(j,k) = CM(j,k)/SQRT[CM(j,j)*CM(k,k)]
c** The squared 95% confidence limit uncertainty in property F({PV(j)})
c  defined in terms of the fitted parameters {PV(j)} is (where the
c  L.H.S. involves  [row]*[matrix]*[column]  multiplication):
c  [D(F)]ˆ2 = [PU(1)*dF/dPV(1), PU(2)*dF/dPV(2), ...]*[CM(j,k)]*
c                              [PU(2)*dF/dPV(1), PU(2)*dF/dPV(2), ...]
c** Externally dimension:  YO, YU and DY  .ge. NDATA (say as MXDATA),
c             P, PU  and  PS  .ge.  NPARM (say as MXPARM),
c             DYDP  with column length MXDATA and row length .ge. NPARM
c             CM   as square matrix with column length  MXPARM
c  Authors: Michael Dulick  &  Robert J. Le Roy, Department of Chemistry
c    U. of Waterloo, Waterloo, Ontario  N2L 3G1.   Version of:  30/08/93
c************************************************************************
      INTEGER I,J,K,L,M,IDF,NDATA,MXDATA,NPARM,MXPARM
      REAL*8  YO(NDATA), YU(NDATA), DY(NDATA), PV(NPARM), PU(NPARM),
     1   PS(NPARM), DYDP(MXDATA,NPARM), CM(MXPARM,MXPARM), SERR,
     2   PX(60), F95(10), TFACT, S, U
      DATA F95/12.7062D0,4.3027D0,3.1824D0,2.7764D0,2.5706D0,2.4469D0,
     1  2.3646D0,2.3060D0,2.2622D0,2.2281D0/
c
           .  .  .
```

```
c***********************************************************************
      SUBROUTINE LLSQFVL(NDATA,NPARM,MXDATA,MXPARM,YO,YU,DY,P,PU,PS,
     1                 CM,SERR)
c**  Program for performing linear least squares fits using orthogonal
c  decomposition of the Design (partial derivative) matrix.
c** This version of the program is designed for the case of very large
c  data sets when it would be inconvenient (or impossible) to store the
c  complete partial derivative matrix at one time, so it is generated
c  row-by-row in calls to USER-SUPPLIED subroutine  DYIDPJ.  For
c  problems involving data set of modest size, it may be more convenient
c  to use program version LLSQF, which assumes that the complete partial
c  derivative array is prepared and brought in as input.
c
c** On entry: NDATA  is the number of data to be fitted (.le.MXDATA)
c             NPARM  the number of parameters to be varied (.le.MXPARM)
c                 If NPARM.le.0 , simply calculate the (weighted RMS
c                 deviation)=SERR  from the input YO(i) & YU(i)’s
c             MXDATA & MXPARM are array dimension parameters (see below)
c                 Internal array sizes currently assume  MXPARM .le. 60
c             YO(i)  are the NDATA ’observed’ data;  for iterative
c                  non-linear fits these are:  [Y(obs,i) - Y(trial,i)]
c             YU(i)  are the uncertainties in these YO(i) values
c
c** On Exit: PV(j)  are the fitted parameter values;  for iterative
c                  non-linear fits these are the parameter changes
c            PU(j) are 95% confidence limit uncertainties in the PV(j)’s
c            PS(j) are ’parameter sensitivities’ for the PV(j)’s, defined
c               such that the RMS displacement of predicted data  due to
c               rounding off parameter-j by PS(j) is .le. SERR/10*NPARM
c            SERR  is the standard error of the fit
c            DY(i) is the array of differences  [YO(i) - Ycalc(i)]
c            CM(j,k)  is the correlation matrix obtained by normalizing
c   variance/covariance matrix:  CM(j,k) = CM(j,k)/SQRT[CM(j,j)*CM(k,k)]
c** The squared 95% confidence limit uncertainty in property F({PV(j)})
c  defined in terms of the fitted parameters {PV(j)} is (where the
c  L.H.S. involves  [row]*[matrix]*[column]  multiplication):
c  [D(F)]^2 = [PU(1)*dF/dPV(1), PU(2)*dF/dPV(2), ...]*[CM(j,k)]*
c                             [PU(2)*dF/dPV(1), PU(2)*dF/dPV(2), ...]
c** Externally dimension:  YO, YU and DY  .ge. NDATA (say as MXDATA),
c             P, PU  and  PS  .ge.  NPARM (say as MXPARM),
c             CM   as square matrix with column length  MXPARM
c  Authors: Michael Dulick  &  Robert J. Le Roy, Department of Chemistry
c    U. of Waterloo, Waterloo, Ontario  N2L 3G1.   Version of:  30/08/93
c***********************************************************************
      INTEGER I,J,K,L,M,IDF,NDATA,MXDATA,NPARM,MXPARM
      REAL*8  YO(NDATA), YU(NDATA), DY(NDATA), PV(NPARM), PU(NPARM),
     1   PS(NPARM),CM(MXPARM,MXPARM), SERR,
     2   PX(60), F95(10), TFACT, S, U, Z1
      DATA F95/12.7062D0,4.3027D0,3.1824D0,2.7764D0,2.5706D0,2.4469D0,
     1  2.3646D0,2.3060D0,2.2622D0,2.2281D0/,Z1/1.d0/
c
     . . .
```