ELSEVIER

Computer Physics Communications 102 (1997) 252-268

Computer Physics Communications

I-NoLLS: a program for interactive nonlinear least-squares fitting of the parameters of physical models

Mark M. Law [1], Jeremy M. Hutson

Department of Chemistry, University of Durham, South Road, Durham DH1 3LE, UK

Received 31 December 1996

Abstract

The I-NoLLS program is a package for carrying out interactive nonlinear least-squares fits to determine the parameters of physical or mathematical models from experimental or other data, under circumstances where automated least-squares procedures are excessively computationally expensive and the user needs interactive control to apply physical insight to the fitting process. The program was developed to facilitate the fitting of molecular potential energy surfaces (PES) to spectroscopic and scattering data, but is also applicable to a variety of other optimization problems. A range of different algorithms adapted to highly nonlinear least-squares problems may be selected. The interactive nature of the code permits rapid and flexible control over the progress of the fit. I-NoLLS is written in a modular way that allows the easy incorporation of new modules for calculating observable quantities from model parameters. The structure of the program allows straightforward parallelisation of the time-consuming property calculations. In pilot applications, I-NoLLS has been interfaced with programs for calculating bound states of Van der Waals complexes, cross sections for molecular scattering processes, and second virial coefficients of gas mixtures. Parallelisation of the property calculations has been achieved using PVM running on a cluster of workstations.

Keywords: Interactive; Nonlinear least-squares; Molecular potential energy surface; Optimization; Model fitting

PROGRAM SUMMARY

Title of program: I-NoLLS

Catalogue number: ADFN

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland

Licensing provisions: Persons requesting the program must sign the standard CPC nonprofit use licence

Computers: The program was developed principally on IBM RS/6000 systems, but should be easily portable to Unix workstations from other manufacturers

Operating systems under which the program has been tested: Unix

Programming language used: FORTRAN 77 with some industry-standard extensions as described in the README file supplied with the program

Memory required to execute with typical data: case-dependent

No. of processors used: case-dependent

Peripherals used: disc files, X-terminal

No. of bytes in distributed program including test data, etc.:

[1] Present and permanent address: Department of Chemistry, University of Aberdeen, Meston Walk, Old Aberdeen AB24 3UE, UK.

386275

Distribution format: uuencoded compressed tar file

Keywords: Interactive, nonlinear least-squares, molecular potential energy surface, optimization, model fitting

Nature of physical problem
I-NoLLS permits interactive least-squares fitting of the parameters of analytical models to physical properties or other data.

Method of solution
A variety of algorithms adapted to nonlinear least-squares optimization are used, including Gauss-Newton [1], Levenberg-Marquardt [2,3] and Singular Value Analysis based [4] algorithms. The interactive nature of the code permits rapid and flexible user control over the progress of the fit.

Typical running time: problem-dependent

Unusual features of the program
The user must supply a subroutine that calculates theoretical values of the quantities to be fitted and their first partial derivatives with respect to the adjustable parameters.

Other required software: The Application Visualisation System (AVS) [5] is required to provide the user interface.

References
[1] P.R. Bevington, Data Reduction and Error Analysis for the Physical Sciences (McGraw-Hill, London, 1969).
[2] K. Levenberg, Quart. Appl. Math. 2 (1944) 164.
[3] D. Marquardt, SIAM J. Appl. Math. 11 (1963) 431.
[4] C.L. Lawson and R.J. Hanson, Solving Least Squares Problems (Prentice-Hall, New Jersey, 1974).
[5] Application Visualisation System computer code, Release 5.0, distributed by Advanced Visual Systems Inc., USA (1993).

LONG WRITE-UP

1. Introduction

The object of this paper is to describe an interactive program for fitting the parameters of a physical model to experimental or other data. The program is called I-NoLLS (an acronym for Interactive Non-Linear Least-Squares and pronounced 'eye-knolls'). The program was developed to deal with problems that arise in the fitting of molecular potential energy surfaces to experimental results from spectroscopic and scattering studies, but it is applicable to a wide range of fitting problems.

The general theory of least-squares fitting is well understood [1-4], and many general-purpose fitting packages are available. However, in our experience these packages are often unsuitable for fitting potential energy surfaces. The special characteristics of the fitting problems that we encounter are the following:

(i) Very expensive calculations are needed to evaluate the observable quantities (physical properties) from a trial set of parameters. Conventional least-squares packages are often written on the assumption that the calculation is cheap, and evaluate the properties for an unnecessarily large number of parameter sets.

(ii) The problems are highly nonlinear, and often poorly determined or underdetermined in mathematical terms. Under these circumstances, an automated algorithm may try to sample unphysical regions of parameter space, and either the program crashes or the fit diverges. Nevertheless, the scientist doing the fitting has physical insight that can distinguish between different proposed parameter sets, and can make choices about the most appropriate parameters to adjust at each stage of the fit.

The I-NoLLS program has been designed to handle fitting problems with these characteristics, and to permit the user to guide the fitting process using physical intuition.

A major innovation in the I-NoLLS program is the introduction of the concept of a *super-Jacobian* matrix containing all the first partial derivatives of the calculated properties with respect to the parameters of the model. The user is able to select a variety of trial submatrices from this matrix and for each submatrix may choose between a variety of different algorithms for calculating least-squares parameter steps.

The structure of the remainder of this paper is as follows. The problems associated with the least-squares fitting of molecular potential energy surfaces to experimental data are outlined in Section 2, the parts of the theory of nonlinear least-squares fitting needed for our discussion are presented briefly in Section 3, the I-NoLLS code is described in Section 4, and some initial applications are discussed in Section 5. Section 6 contains some concluding remarks.

2. Fitting molecular potential energy surfaces

This section outlines the physical problem that motivated the development of the I-NoLLS program. Users who wish to use the code for different applications may skip to Section 3.

Intermolecular and intramolecular potential energy surfaces (PESs) are of central importance in molecular physics and many areas of physical chemistry [5]. The theory needed to calculate molecular properties (such as energy levels and scattering cross sections) from potential energy surfaces is highly developed [6-13], though technically demanding and computationally expensive. However, for most systems the limiting factor is that we do not know the potential surface accurately enough for property calculations to be reliable. Although ab initio electronic structure calculations can provide a great deal of information on potential energy surfaces, they are not yet cheap enough or accurate enough for many purposes. Because of this, a great deal of work has been done using *empirical* PESs determined from experimental data. The availability of accurate empirical PESs for prototype systems is also crucial in evaluating and developing new ab initio methods.

There are several approaches that have been used to address the problem of determining molecular PESs from experimental data. These may be classified loosely as (i) methods based on determining the parameters of an assumed functional form, and (ii) methods based on direct inversion of the experimental data, often to obtain a pointwise potential surface [14,15]. In either case, it is often desirable to include some information from ab initio electronic structure calculations or other theoretical considerations in order to determine regions or features of the PES that are not well determined by experimental data.

The present paper concentrates on methods for determining the parameters of an assumed functional form. Although this has often in the past been done simply by manual adjustment of selected parameters to obtain agreement with a particular set of data, such a procedure is not very satisfactory because it does not give a clear indication of the uncertainties in the resulting potential. It is much better to use a proper nonlinear least-squares procedure to determine the optimum parameters, along with statistical information on their uncertainties and the correlations between them[2].

The major problems encountered in applying least-squares methods to the determination of molecular PESs are those arising from severe nonlinearity and the fact that the available data are often insufficient in themselves to provide a unique solution for the potential parameters. The nonlinearity requires that the solution procedure is *iterative* and can make it computationally very expensive, while the underdetermined nature makes it necessary for the scientist to apply physical insight to choose between different possible solutions as the fit proceeds. Even if the problem is not strictly underdetermined, the parameters are often highly correlated, and once again physical insight is needed to obtain a satisfactory solution. It is not always possible to express the physical insight in terms of explicit constraints on the parameters to be fitted.

In the last decade, reliable empirical potential energy surfaces have been obtained by least-squares fitting for a variety of molecular systems [18]. In particular, high-resolution spectroscopic data have been shown to provide very detailed information on potential energy surfaces. Highly accurate PESs have been determined for prototype systems such as Ar-H_2 [19], Ar-HF [20], Ar-HCl [21], and HCN [22]. Most of these have been determined using standard automated least-squares procedures, though the automated procedures proved poorly adapted to the special problems encountered. Fits have also been described for more complex intermolecular systems such as Ar-CO_2 [23,24], Ar-OH [25,26], Ar-H_2O [27,28], Ar-NH_3 [29], $(HF)_2$ [30] and $(HCl)_2$ [31], but it is inevitable that for such larger systems the data sets are less complete and greater problems due to correlation and underdetermination are encountered. For larger chemically bound systems such as CH_3F [32], it is usually possible to determine only the harmonic force field from spectroscopic data.

Some programs for fitting PESs to experimental data have been published, of which the most notable

[2] The least-squares procedure is not the only method available for parameter estimation. However, it is the most widely used in the physical sciences. For discussions of its relationships to other methods and its advantages we refer the reader to Refs. [16,17]. We note here only that least 23 squares affords an unbiased, minimum-variance solution and that it is a maximum likelihood estimator under the common assumption that the measurement errors are independent and normally distributed.

are ASYM20 [1] and CFF-GOPT [33,34]. ASYM20 is a program designed to perform refinement of harmonic force fields, with some modest interactive control implemented. CFF-GOPT includes a graphical user interface and is adapted to the problem of developing force fields for molecular mechanics calculations using built-in potential energy functions. In comparison with these codes, I-NoLLS permits more flexible interaction with the user, and offers a range of different algorithms adapted to highly nonlinear least-squares problems.

The general approach taken in our work is that theoretical values of molecular properties are calculated for a given trial potential energy function using specialised codes. I-NoLLS compares the calculated and experimental values, and adjusts the parameters so as to reduce the residual discrepancies. The I-NoLLS user is encouraged to choose between various possible steps in parameter space on physical grounds. The whole process is repeated iteratively until convergence is achieved or the user decides to stop.

3. Nonlinear least-squares fitting

There is no single least-squares algorithm that may be regarded as universally the most effective. Different algorithms are suited to different problems, and indeed to different stages in a given problem. The best algorithm depends on the computational expense of calculating molecular properties, the availability of analytic derivatives, the feasibility of parallelisation, the degree of nonlinearity of the model, and so on. Commonly used algorithms include those based on Gauss–Newton [3], Levenberg–Marquardt [35,36], singular value analysis (SVA) [37], steepest descent [3] and iteratively reweighted [38] steps. All of these require the calculation of the *Jacobian* matrix of first partial derivatives of the calculated properties with respect to the parameters of the model (potential energy) function.

Since all of the algorithms listed above have been thoroughly described and analysed in the literature, only a brief outline of the main features of the first three in the list above (which have been implemented in the current version of the I-NoLLS program) will be given here.

We define the least-squares problem as

$$\text{minimise} \quad \chi^2 = \sum_{i=1}^{n} \left[\frac{y_i^{\text{obs}} - y_i^{\text{calc}}(p_1 \cdots p_m)}{\sigma_i} \right]^2, \tag{1}$$

where y^{obs} is the n-vector of observed properties, y^{calc} is the corresponding vector of theoretically calculated values, σ_i is the uncertainty assigned to data point i and p is the m-vector of adjustable parameters.

To obtain an estimate of how to improve a trial vector p of adjustable parameters, we first express y^{calc} as a Taylor series expansion about the current parameter vector,

$$y^{\text{calc}}(p + x) = y^{\text{calc}}(p) + Jx + \cdots, \tag{2}$$

where x is a small change in the parameter vector. The $n \times m$ matrix J is the *Jacobian* matrix of first partial derivatives of the calculated properties with respect to the model parameters,

$$J_{ij} = \frac{\partial y_i^{\text{calc}}}{\partial p_j}. \tag{3}$$

We also define the vector of differences

$$d = y^{\text{obs}} - y^{\text{calc}}(p) = y^{\text{obs}} - y(p). \tag{4}$$

For a weighted least-squares fit, we define a diagonal matrix G with nonzero elements

$$G_{ii} = \frac{1}{\sigma_i}. \tag{5}$$

Some authors express this in terms of a weight matrix $W = G^2$.

If we assume that the model function $y(p)$ is locally linear in p, then we may neglect all but the first two terms in the expansion (2) and reexpress problem (1) as

$$\text{minimise} \quad \|Ax - b\|_2^2, \tag{6}$$

where $A = GJ$ and $b = Gd$ and the notation $\|.\|_2$ denotes the Euclidean length of a vector. Our objective is to find an optimum vector x describing a step through parameter space.

3.1. Singular Value Decomposition

Several of the most useful least-squares algorithms may be expressed [37] in terms of the Singular Value Decomposition (SVD) of the rectangular matrix A,

$$A = UKV^{\text{T}}, \tag{7}$$

where U and V are $n \times n$ and $m \times m$ orthogona matrices, respectively, and, for $n > m$,

$$K = \begin{pmatrix} S \\ 0 \end{pmatrix}. \tag{8}$$

S is a diagonal matrix of dimension m whose elements are the singular values of A and are arranged in descending order,

$$s_1 \geq s_2 \geq \cdots \geq s_m \geq 0. \tag{9}$$

Expressing A in terms of its SVD, Eq. (7), and multiplying from the left by U^T, problem (6) may be transformed into

$$\text{minimise} \quad \left\| \begin{pmatrix} S \\ 0 \end{pmatrix} q - g \right\|_2^2, \tag{10}$$

where $q = V^T x$ and $g = U^T b$. It can be shown [37] that Eq. (10) has a solution vector q with components

$$q_j = \frac{g_j}{s_j}. \tag{11}$$

Hence Eq. (6) has the solution $x = Vq$. This parameter step corresponds to the well-known Gauss–Newton algorithm [3].

It should be noted that y^{calc} is nonlinear with respect to the adjustable parameters, so that the process of finding a new Jacobian matrix and generating a new least-squares step must be repeated iteratively until it converges on a minimum in the sum of squares. Even then, there is no guarantee that the point that has been located is the *global* minimum.

3.2. Parameter scaling

In physical applications, different elements of the Jacobian matrix often have different dimensions (or units). However, in order for a singular value decomposition to be meaningful, either the matrix A must be dimensionless or all its elements must have the same units. The dimensions of the data points are removed through the matrix G, because the uncertainty in each data point has the same dimensions as the quantity itself. However, the choice of scaling for the parameters is to some extent arbitrary: possible scaling factors to reduce the parameters to dimensionless form include (i) the units themselves[3] and (ii) an estimate of the uncertainty in the parameter, such as its 95% confidence limit.

3.3. Parameter correlation

A general problem associated with multidimensional least-squares problems is that the parameters are *correlated*. It is quite possible for a minimum in the sum of squares to exist such that a small change in any one parameter by itself causes a substantial increase in the sum of squares, but that nevertheless there are other directions in parameter space (corresponding to linear combinations of the parameters) which result in a much smaller increase in the sum of squares. Under these circumstances, the parameters are said to be *correlated*, and it is important to know about such correlations. A commonly-used simple measure of correlation is the *correlation matrix* C,

$$C = D^{-1} \Theta D^{-1}, \tag{12}$$

where Θ is the *variance-covariance matrix* given by

$$\Theta = (A^T A)^{-1} = VS^{-2}V^T, \tag{13}$$

and D is a diagonal matrix with elements given by $D_{jj} = \Theta_{jj}^{1/2}$. Off-diagonal elements of the correlation matrix close to ± 1 indicate that the ith and jth adjustable parameters are highly correlated. Parameter uncertainties are also affected by correlation, and this must be taken into account when computing the uncertainties in quantities derived from the parameters. Further details may be found in Ref. [17].

The correlation matrix detects only *linear* dependencies between *pairs* of parameters. Correlations involving more than two parameters may also exist. The SVD approach affords a more general method of detecting such correlations. Associated with each singular value s_j of the matrix A is a direction in parameter space (defined by the corresponding column of V). The singular directions are mutually orthogonal and in contrast to the original parameters p are *uncorrelated* with one another. Each component q_j in

[3] Some authors ignore the question of parameter scaling, and claim to carry out an unscaled SVA. This usually means that they treat the physical quantities in the Jacobian as though they were pure numbers, which implicitly amounts to scaling according to the physical units as in (i) above.

Eq. (11) reduces the sum of squares function by an amount g_j^2. The largest singular values correspond to the best-determined directions in parameter space, and the smallest singular values to the least-determined directions. In the limit $s_1 \gg s_2$, the direction associated with the largest singular value corresponds to the direction of steepest descent.

3.4. *Avoiding long steps in parameter space*

As mentioned above, the parameter step $x = Vq$ corresponds to the Gauss-Newton algorithm [3]. However, it must be remembered that this step will reach the actual minimum in the sum of squares only if the problem is linear, i.e. if the Jacobian is independent of the parameter values. In nonlinear problems far from the minimum, the Gauss-Newton step is often far too long for the linearisation to be valid. The Gauss-Newton algorithm can then produce a parameter set that gives a very poor fit to the data, or in drastic cases is unphysical.

One common modification to alleviate the problems of nonlinearity is simply to multiply the Gauss-Newton step by a reduction factor f, with $f < 1$ [1]. This may produce a worthwhile improvement in the quality of fit without moving so far that the linearization fails. However, it is often better to select a parameter step composed of only the first k components of q (corresponding to the k best-determined singular directions). Such a step does not actually minimise the sum of squares of errors in the linearised problem, but often produces a large reduction for a relatively short step in parameter space. It therefore reduces the likelihood of straying outside the near-linear region of parameter space or reaching an unphysical region of parameter space. This approach also makes it straightforward to handle *underdetermined* least-squares problems, including the case $m < n$. Ref. [39] gives further discussion of this topic in the context of 'snowball' fitting.

3.5. *Levenberg-Marquardt algorithm*

Another approach that is commonly employed to improve the conditioning of the least-squares problem and to reduce the length of the parameter step is the Levenberg-Marquardt approach [35,36]. This may conveniently be expressed in terms of the SVD of the matrix A by modifying Eq. (6) to read

$$\text{minimise} \quad \left\| \begin{pmatrix} A \\ \lambda I_m \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2, \tag{14}$$

where I_m is the identity matrix and λ is a positive scalar. The Levenberg-Marquardt approach thus adds m rows to the Jacobian and data matrices, in effect adding the current parameter values as data points with a weighting determined by λ. Using the SVD of A as before and applying additional orthogonal transformations (Givens rotations) to eliminate the elements of λI_m from the left-hand side gives an equivalent problem [37],

$$\text{minimise} \quad \left\| \begin{pmatrix} S^{(\lambda)}_{m\times m} \\ 0_{n\times m} \end{pmatrix} q^{(\lambda)} - \begin{pmatrix} g^{(\lambda)} \\ h^{(\lambda)} \end{pmatrix} \right\|_2^2, \tag{15}$$

where

$$g_j^{(\lambda)} = \begin{cases} g_j s_j/(s_j^2+\lambda^2)^{1/2}, & j = 1,\dots,m, \\ g_j, & j = m+1,\dots,n, \end{cases} \tag{16}$$

$$h_j^{(\lambda)} = -g_j\lambda/(s_j^2+\lambda^2)^{1/2}, \quad j = 1,\dots,m, \tag{17}$$

and $S^{(\lambda)}$ is a diagonal matrix with nonzero elements

$$s_j^{(\lambda)} = (s_j^2+\lambda^2)^{1/2}, \quad j = 1,\dots,m. \tag{18}$$

The solution vector, $q^{(\lambda)}$, of Eq. (15) has components

$$q_j^{(\lambda)} = \frac{g_j^{(\lambda)}}{s_j^{(\lambda)}} = \frac{g_j s_j}{s_j^2+\lambda^2} = q_j^{(0)}\frac{s_j^2}{s_j^2+\lambda^2}, \quad j = 1,\dots,m. \tag{19}$$

Each component of $q^{(\lambda)}$ reduces the sum of squares of errors by

$$g_j^2\left[1 - \left(\frac{\lambda^2}{s_j^2+\lambda^2}\right)^2\right], \quad j = 1,\dots,m. \tag{20}$$

Eqs. (14)-(20) may be regarded as a generalisation of the SVD-based equations, with $(s_j^2+\lambda^2)^{1/2}$ replacing s_j. Setting $\lambda = 0$ recovers the Gauss-Newton step, and its SVD components, while a value of λ that is large compared with the largest singular value of A yields a very short parameter step along the direction of steepest descent. The former is usually the most appropriate for nearly linear problems or near to a solution, whilst the latter leads to more efficient iterations for highly nonlinear problems far from the solution. It can be seen from Eq. (19) that increasing λ preferentially attenuates parameter changes along the most poorly determined directions (that is, those associated with the smallest singular values). Even within the Levenberg–Marquardt approach, it is often advantageous to include only the first k components of $q^{(\lambda)}$, corresponding to the k best-determined directions in parameter space.

The most appropriate value for λ depends on the degree of nonlinearity of the model function. Although it is often desirable to choose λ manually, automated procedures are also available for choosing values of λ for a sequence of iterations [40].

For further discussion of the SVA and Levenberg–Marquardt algorithms and the relationship between them, see Ref. [37].

3.6. Parameter constraints

It is often desirable to constrain a least-squares problem to a subspace of the full m-dimensional parameter space. A set of t linear parameter constraints may be specified in the form

$$Tp = \tau, \qquad (21)$$

where T is a $t \times m$ matrix of coefficients and τ a t-vector of scalars. Such constraints may be taken into account in the calculation of the least-squares parameter step, x, using the method described by Gill et al. [4]. Briefly, the problem (6) is recast as

$$\text{minimise} \quad \|AZx - b\|_2^2, \qquad (22)$$

where Z is an $m \times (m-t)$ matrix whose columns are orthogonal to the rows of T. This procedure can be used to impose constraints on the end point of a step even when the starting point does not conform to the constraints.

Another possibility is that the user may wish to restrict the parameter values to a certain region of parameter space, without imposing rigid constraints. This may be done simply by extending the vector of "observable" properties y^{obs} to include the preferred values for the parameters concerned, each with an associated uncertainty. This has been termed the "method of predicate observations" [41]. It both allows the user to build in expectations about parameter values based on factors external to the fit (such as theoretical values or prior experience) and may alleviate problems associated with parameter correlation. Each parameter may be viewed as being "tied" to its predicate value by a spring whose stiffness is determined by the associated uncertainty.

The method of predicate observations may easily be generalised to allow any *function* of the parameters (linear or nonlinear) to be treated as an "observable".

4. I-NoLLS program

4.1. Philosophy

The basic philosophy of the I-NoLLS program is that, at each iteration, the user requests a *super-Jacobian* matrix, which contains all the data that might be included in the fit and all the parameters that might be varied. After inspecting the current quality of fit, the user can point and click with a mouse to select a *sub-Jacobian* containing a subset of the data and parameters for inclusion in the current least-squares step. For example, if some of the parameters cannot yet be reliably determined from the data, or if the forwards calculation fails for some of the properties, it may be desirable simply to click them out of the fit and proceed with the remainder. Once the sub-Jacobian is chosen, the user can experiment with different algorithms for calculating the next step, and can select one that seems physically acceptable. If at this stage the full parameter space is found to be too correlated for a sensible step to be identified, the user may select a new (and usually smaller) set of adjustable parameters for inclusion in the sub-Jacobian, without needing to recalculate any Jacobian matrix elements. The user may also choose to include one or more linear parameter constraints. Finally, after a promising step has been chosen, I-NoLLS carries out a single calculation of the properties for the new set of parameters, and in the light of the results the user can choose either to accept the step (and perhaps proceed to calculate a new super-Jacobian) or to reject the step and try to find a better one based on the old super-Jacobian.

4.2. Program modules

The I-NoLLS package consists of a main module (executable `main.g`) and four separate interactive program modules `results`, `ptoggl`, `chsalg` and `chsflm` (executables `results.g`, `ptoggl.g`, `chsalg.g` and `chsflm.g`). The interactive modules are initiated as separate processes as required by the corresponding `main.g` subroutines. `results` is used to view the least-squares statistics and toggle on or off various interactive stages. `ptoggl` is used to choose which data and model parameters to include in the super-Jacobian and sub-Jacobian matrices and which constraints to include. `chsalg` is used to choose between different least-squares algorithms and scalings, and `chsflm` allows the user to consider and select suitable SVA and/or Levenberg–Marquardt parameter steps.

All the interactive modules communicate with the `main` process via disc files as shown in Fig. 1. `main` performs all the important input/output operations and provides most of the numerical functionality needed for the least-squares fitting process. It calls a user-supplied subroutine CALCYJ to evaluate the vector $y^{\rm calc}$ of calculated properties and/or the Jacobian matrix J corresponding to the current parameter vector p.

4.3. User interface

I-NoLLS uses a graphical user interface to give the user rapid and flexible control over the progress of the fit. In the present implementation, we have adopted the Application Visualisation System (AVS) [42] to provide the interactive functionality. AVS provides a library of *module control widgets*, such as dials, toggle icons and type-in dialog boxes, all controlled by mouse clicks. In our applications, we have used a colour X-terminal as the display device. AVS also offers advanced graphics capabilities that allow for the possible future development of sophisticated visualisation functionality for the complex data sets associated with least-squares fitting.

The main module `main.g` is implemented in FORTRAN 77 as an AVS 'co-routine' (represented in AVS by an icon labelled 'I-NoLLS'). This calls the main least-squares subroutine ILS. The latter is written entirely independently of AVS and makes calls to all the other subroutines and program units.

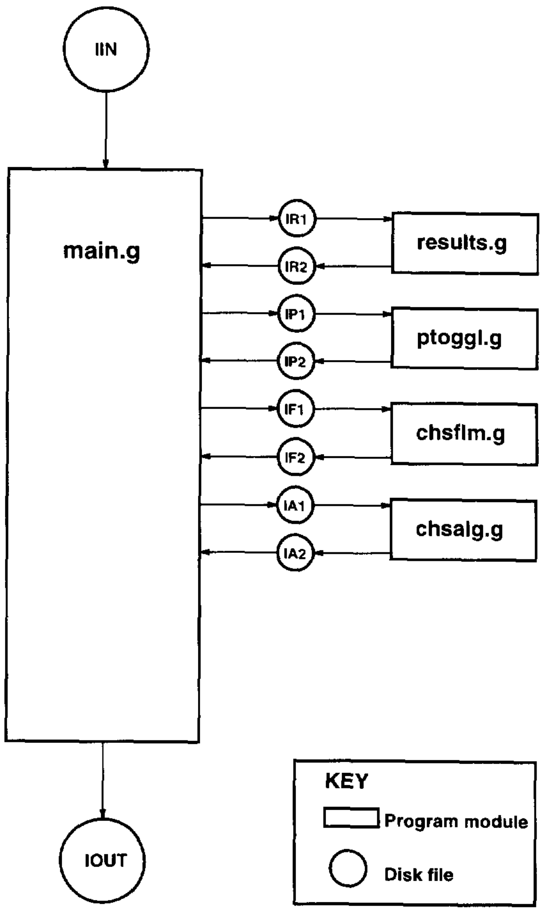

Fig. 1. Program module flow diagram. Disc files are indicated by FORTRAN unit numbers. Note that the modules `results`, `ptoggl`, `chsalg` and `chsflm` may be invoked many times by the module `main`.

The user interaction modules are implemented as four separate AVS 'subroutine' modules, also in FORTRAN 77.

The modularity of I-NoLLS ensures that the interactive components provided by AVS could easily be replaced by equivalent routines based on some other widget library or visualisation system, or could even be replaced by an entirely different method of driving the least-squares code (such as an expert system).

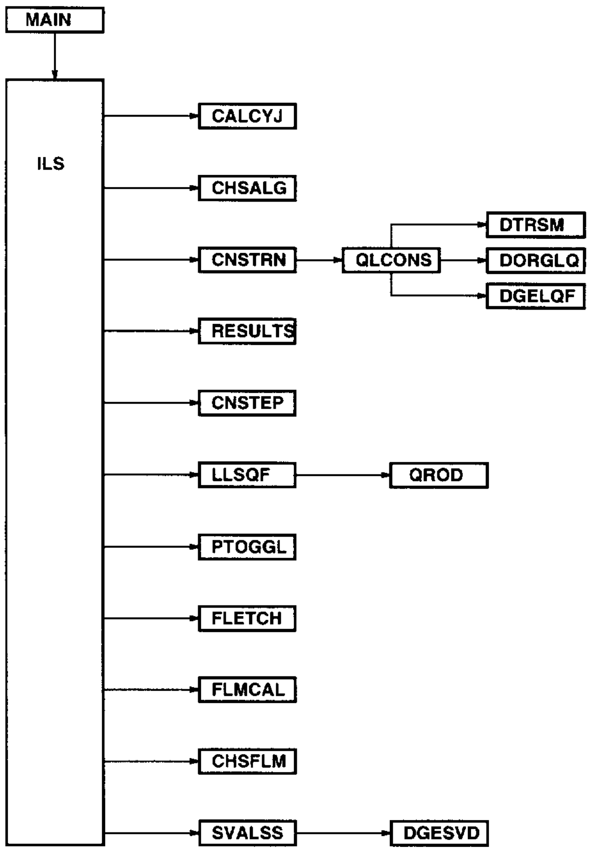

Fig. 2. Structure of module main. A number of service routines have been omitted.

4.4. Subroutine structure

The subroutine structure of `main.g` is shown in Fig. 2. Brief descriptions of the functions of the most important subroutines are given in Table 1. The user-supplied routine CALCYJ is described in detail in subsection 4.6.

Of the interactive program units, `chsflm.g` is the only one which has any significant subroutine structure, as shown in Fig. 2.

4.5. Description of input

The I-NoLLS program follows the convention that variables with names beginning with A–H and O–Y are 8-byte reals, those with names beginning with I–N are integers, and those with names beginning with Z are logicals.

The program runs principally under interactive control using a mouse or similar pointing device. Nevertheless, the basic information about the properties to be fitted, the parameters to be determined, and any linear constraints to be applied is read from a formatted file (stream IIN with the filename extension '.data'). The contents of the data file are as follows:

Record 1: TITLE0 (A72) 72-character title.

Record 2: NPARM0, NDATA0, NJCP0, NRCP0, NJCY0, NRCY0, NCONS0, IPRINT (free format).

NPARM0	Number of parameters to be read.
NDATA0	Number of data points to be read.
NJCP0	Number of INTEGER controls associated with each parameter.
NRCP0	Number of REAL*8 controls associated with each parameter.
NJCY0	Number of INTEGER controls associated with each data point.
NRCY0	Number of REAL*8 controls associated with each data point.
NCONS0	Number of constraints to be read.
IPRINT	Print level.
	= 0, no output to *.out file.
	= 1, normal output to *.out file.
	= 2, beginner's help.
	> 2, extensive debugging output, not described here.

Record 3: NITER (free format).

NITER is reserved for future expansion; for this version, set NITER to -1 and terminate the card with the '/' character.

Record 4 - Record 3+NPARM0: PLABEL (A16), P0, ICP0, (JCP0(I), I = 1, NJCP0), (RCP0(I), I = 1, NRCP0) (free format after 16-character label).

These records specify the parameters of the model function. Each parameter has associated arrays of integer and real control parameters; these control parameters are not used by I-NoLLS itself, but are passed to CALCYJ and may be used as desired by the specialist codes for calculating properties.

Table 1
I-NoLLS subroutines and their functions

ILS	Principal I-NoLLS subroutine: reads input data and handles calls to major subroutines.
RESULTS	Views the fit and select between running modes.
PTOGGL	Toggles data/parameters in/out of fit.
CHSALG	Interactive choice of least-squares algorithm.
CHSFLM	Interactive choice of Levenberg-Marquardt parameter.
CALCYJ	User-supplied routine to calculate properties and their derivatives with respect to the model parameters.
LLSQF & QROD	Calculate Gauss-Newton parameter step[a].
SVALSS	Calculates and writes out the set of SVD parameter steps or the corresponding steps modified by a nonzero Levenberg-Marquardt parameter.
DGESVD	LAPACK routine [48] to calculate the singular value decomposition used by SVALSS.
FLMCAL & FLETCH	Provide an automatic selection of the Levenberg-Marquardt parameter using an algorithm proposed by Fletcher [40].
CNSTRN & QLCONS	Adjust vector of differences and Jacobian to take constraints into account.
CNSTEP	Adjusts parameters with linear constraints.
DGELQF & DORGLQ	LAPACK routines [48] to compute QL factorisation used by QLCONS.
DTRSM	Level 3 BLAS routine [49] called by QLCONS to solve matrix equation for constrained parameter space.

[a] The LLSQF and QROD subroutines were written by M. Dulick, adapted by R.J. Le Roy (University of Waterloo) and modified by the present authors.

PLABEL	String label for parameter.
P0	Initial trial parameter value.
ICP0	= 1, parameter is initially included in fit.
	= 0, parameter is initially excluded from fit.
JCP0(I)	Array of INTEGER control parameters used by CALCYJ.
RCP0(I)	Array of REAL*8 control parameters used by CALCYJ.

Record 4+NPARM0 - Record 3+NPARM0+NCONS0: CLABEL (A16), ICC0, (CONSTM(I), I = 1, NPARM0), BR (free format after 16-character label).

These records specify any linear parameter constraints that *may* be applied in the fitting process. Note that the user has the opportunity to toggle these constraints in and out of the fit interactively as the fit proceeds.

CLABEL	String label for constraint.
ICC0	= 1, constraint is initially included in fit.
	= 0, constraint is initially excluded from fit.
CONSTM(I)	Coefficients of the parameters in the constraint (elements of the constraint matrix T).
BR	Constant term in constraint (element of vector τ).

Record 4+NPARM0+NCONS0 - Record 3 +NPARM0+NCONS0+NDATA0: YLABEL (A16), ICY0, (JCY0(I), I = 1, NJCY0), (RCY0(I), I = 1, NRCY0) Y0, UY0 (free format after 16-character label).

These records specify the experimental data that are to be fitted. Each data point has associated arrays of integer and real control parameters; these control parameters are not used by I-NoLLS itself, but are passed to CALCYJ and may be used as desired by the specialist codes for calculating properties.

YLABEL	String label for datum (observable quantity).
ICY0	= 1, datum is initially included in fit.
	= 0, datum is initially excluded from fit.
JCY0(I)	Array of INTEGER control parameters used by CALCYJ.
RCY0(I)	Array of REAL*8 control parameters used by CALCYJ.
Y0	Value of datum.
UY0	Uncertainty in datum.

4.6. *Specification of subroutine* CALCYJ

The subroutine CALCYJ must be supplied by the user. Its purpose is to calculate the properties corresponding to a trial set of model parameters and the super-Jacobian matrix of their derivatives with respect to the parameters. In complicated applications, CALCYJ may start up other programs to perform the property calculations, rather than containing all the necessary code itself.

Any predicate observations can be dealt with very simply by including appropriate code in CALCYJ.

The calling sequence for CALCYJ is

```
SUBROUTINE CALCYJ(IY1, IJ1, NPARM0, NDATA0, P0, ICP0, RCP0, Y0, ICY0, RCY0, Y1, DYDP1, NPARM1, NDATA1, ZFNERR, JCP0, JCY0)
```

NPARM0, NDATA0, NPARM1 and NDATA1 are integer variables, and ZFNERR is a logical variable. The remaining parameters are arrays, which must be dimensioned the same as in the calling routine (IMAIN). The dimensions are

```
P0(MXPARM)
ICP0(MXPARM,MXICP0)
RCP0(MXPARM,MXRCP0)
Y0(MXDATA)
ICY0(MXDATA,MXICY0)
RCY0(MXDATA,MXRCY0)
Y1(MXDATA)
DYDP1(MXDATA,MXPARM)
JCP0(MXPARM,MXJCP0)
JCY0(MXDATA,MXJCY0)
```

All the dimensions are defined as FORTRAN PARAMETERS in the include file ls.par.h, which is supplied as part of the I-NoLLS distribution.

The variables NPARM0 and NDATA0 and the arrays P0, ICP0, RCP0, Y0, ICY0, RCY0, JCP0 and JCY0 have the same significance as described in Subsection 4.5 above (except that they refer to *current* values, which may have been changed during the course of the fit so far). These together with the variables IY1 and IJ1 must be left unchanged by CALCYJ. The variables NPARM1, NDATA1 and ZFNERR and the arrays Y1 and DYDP1 must be calculated and returned by CALCYJ.

It may be necessary for CALCYJ to read from the data file or write to the output file. If so, it may incorporate the COMMON block

```
COMMON /A1/ IIN, IOUT, IPRINT
```

where IIN, IOUT and IPRINT are INTEGER variables (defined in subroutine ILS). IIN and IOUT are the FORTRAN unit numbers of the input and output files, and IPRINT is the (input) print level control. Channel IOUT should be used for debugging output or error details.

The usage of the remaining parameters in the calling sequence is as follows:

IY1	Integer flag (input): = 1, Y1 array (calculated properties) must be returned. = 0, Y1 array must be left unchanged.
IJ1	Integer flag (input): = 1, DYDP1 array (super-Jacobian) must be returned. = 0, DYDP1 array must be left unchanged.
Y1	Array of dimension MXDATA (output): If IY1 is 1 on input, the elements of Y1 must be returned containing the values of the properties calculated for the trial set of parameters supplied. Note that *all* NDATA0 elements of Y1 must be returned, even if the data point concerned is not currently included in the super-Jacobian.
DYDP1	Array of dimension (MXDATA, MXPARM) (output): If IJ1 is 1 on input, DYDP1 must be returned containing the super-Jacobian matrix, i.e. the matrix of first partial derivatives of the calculated properties with respect to the model parameters. Note that an element of DYDP1 needs to be returned only if both the data point and the parameter concerned are required in the super-Jacobian, as specified by ICP0 and ICY0. DYDP1 is returned packed, so that only elements up to DYDP1(NDATA1,NPARM1) are occupied; there are no rows and columns containing zeroes for data and parameters that are not in the super-Jacobian.
NPARM1	Integer variable (output): If IJ1 is 1 on input, NPARM1 must be returned as the number of model parameters included in the super-

Jacobian (that is, the number of columns of DYDP1 returned). CALCYJ can calculate this by scanning the array ICP0 to count the number of 1s.

NDATA1 Integer variable (output):
If IJ1 is 1 on input, NDATA1 must be be returned as the number of data points included in the super-Jacobian (that is, the number of rows of DYDP1 returned). CALCYJ can calculate this by scanning the array ICY0 to count the number of 1s.

ZFNERR Logical flag (output):
=.TRUE. if an error occurred in CALCYJ.
=.FALSE. if no error occurred in CALCYJ.
If the value .TRUE. is returned, then I-NoLLS displays an on-screen warning ERROR IN CALCYJ, which the user must acknowledge before proceeding. I-NoLLS then carries on as normal; it is up to the user to look in the output file (unit IOUT) for any (user-defined) error details written by CALCYJ.

4.7. User interaction and output

The program is started by opening the AVS Network Editor and pulling down the icon labelled `I-NoLLS`. Next, the user selects a data file (as described in Section 4.5) from a menu listing all files with extension '.data' in the AVS data directory. The program then performs a single property calculation at the point in parameter space specified in the data file.

As I-NoLLS proceeds, it calculates key sets of statistics which the user may wish to consider in assessing the desirability of a particular least-squares step. These results may be viewed interactively (as described below) and are also recorded in the output file. The latter inherits the filename of the input file with the extension '.data' replaced by '.out'. Throughout the output listing, the user-defined parameter, data and constraint string labels are attached to the often large and complex tables and matrices.

Most of the main user interaction menus include a context-sensitive HELP button, which opens a window displaying information about the options available at that stage.

After the initial property calculation, I-NoLLS enters the `results` module. The `results` display includes the variance $\sigma^2 = \chi^2/(n-m)$, which summarizes the overall quality of fit[4]. To see a more detailed breakdown of the results, the user may pick the `View Output` button, which starts up a text editor (vi by default) to view the main output file. At this stage, the output file contains listings of the observed and calculated values of each property and the corresponding weighted and unweighted residuals (observed–calculated values).

The `results` module also displays a number of switches that control the extent to which I-NoLLS will request user interaction as the fit proceeds. Each switch may be toggled between `on` (highlighted) and `off` (unlit) states with mouse clicks. The `Sup-Jac Param`, `Sup-Jac Data`, `Sub-Jac Param` and `Sub-Jac Data` switches control whether the user will be prompted to reselect the fitting parameters and data to be included in the super-Jacobian and sub-Jacobian matrices each time the program reaches the appropriate point. The `Constraints` switch controls whether the user is prompted to reselect which linear constraints are to be applied, while the `Algorithms` switch controls whether the user is prompted to reselect a least-squares algorithm each time the program is ready to generate a new least-squares step.

The user may leave the `results` module by picking either `Choose/Calculate Jacobian` or `Do N cycles`. The latter simply executes N least-squares cycles without further interaction, using automatically adjusted steering parameters; the value of N is taken from the adjacent dialog box. If the user picks `Choose/Calculate Jacobian`, I-NoLLS starts up the `ptoggl` module to allow the user to specify the super-Jacobian matrix (provided the `Sup-Jac Param` and/or `Sup-Jac Data` switches are set on). `ptoggl` displays a list of parameters and/or data, each represented by a button showing the user-supplied string that identifies the quantity concerned: mouse clicks are used to highlight the quantities that are to be included in the super-Jacobian. Once the selection is

[4] Note that the list of parameters and data included in the fit, and hence the interpretation of σ^2, may change under user control as the fit proceeds.

complete, the user picks Done to proceed to the next stage.

Once the parameters and data have been selected, I-NoLLS proceeds to calculate the super-Jacobian. This is usually the most expensive part of the calculation. The complete super-Jacobian is written to the main output file. I-NoLLS then enters ptoggl again (twice) for the user to select a sub-Jacobian, and once more for the user to pick the linear parameter constraints to be included (if any were included in the data file). Again, these calls to ptoggl are suppressed if the corresponding switches are set off.

The program then enters the chsalg module, which allows the user to choose a least-squares algorithm to be applied in the next step. The present version of I-NoLLS offers three choices: Gauss--Newton, SVA-LM scaling 1 and SVA-LM scaling 2. The latter two use different scalings to reduce the Jacobian matrix to dimensionless form before the singular value decomposition. Scaling 1 simply scales by the units in which the parameters were input: in other words, it treats the values in the Jacobian matrix as though they were pure numbers, not physical quantities. Scaling 2 scales the parameters by their 95% confidence limits instead.

The Gauss–Newton least-squares algorithm has no steering parameters, so the program proceeds directly to the next call of results as described below. However, if one of the SVA-based algorithms is selected in chsalg, I-NoLLS next enters the module chsflm, which allows the user to experiment with the control parameters of the Singular Value Analysis and Levenberg–Marquardt algorithms. chsflm computes and maintains an interactive display of several matrices and other quantities that are valuable in solving difficult least-squares problems. These include the singular values s_j of the matrix A, the corresponding left-hand and right-hand singular vectors (i.e., the matrices U and V^T, respectively) and the rank of A[5]. chsflm also displays the ratio of the largest and smallest (nonzero) singular values, which is a good measure of the overall correlation of the least-squares problem.

chsflm gives the user the opportunity to experiment interactively with different values of the Levenberg–Marquardt parameter λ. When a value of λ is typed in to an on-screen dialog box, tabulations are displayed showing the length of the resulting parameter step, the predicted (linearised) reduction in σ^2 and the parameter changes associated with each SVA component of the solution vector q. The corresponding predicted residuals are also given. All these quantities are updated interactively when the user alters the on-screen value of the Levenberg–Marquardt parameter.

chsflm also allows the user to choose a parameter step composed of only the k best-determined directions in parameter space, corresponding to the k largest singular values of A.

Once promising values of λ and k have been chosen, the user exits from chsflm. The proposed parameter values, their changes and 95% confidence limits, and the correlation and variance–covariance matrices are written to the output file. I-NoLLS then reenters results, which displays the predicted value of the variance σ^2 and offers the opportunity to view the output file. The user may also change the values of the switches that control user interaction. At this stage, the user may either reject the step and return to seek a new one based on the old super-Jacobian (but perhaps using a new sub-Jacobian or another least-squares algorithm), or may pick Try parameter step to accept the proposed step provisionally and carry out a calculation of the actual properties and residuals at the new point in parameter space. results also offers the option to reduce the length of the step by multiplying it by a reduction factor if desired. If the step is accepted, the actual calculated values are tabulated in the output file, along with associated quantities such as the original observed values, the (observed–calculated) residuals, the uncertainties and the weighted squares of the residuals.

After the property calculation, I-NoLLS reenters results and the user has a final opportunity to inspect the results and either confirm acceptance of the step or return to seek a new step based on the old parameter values and super-Jacobian. If the user picks Accept parameter step, I-NoLLS will proceed to another iteration, calculating a new super-Jacobian at the new point in parameter space. Alternatively, the user may

[5] As usual, the parameters and data are identified where possible by the corresponding user-supplied strings. However, when parameter constraints are being applied, the parameter space is subject to an additional linear transformation which complicates the interpretation of the singular vectors. In I-NoLLS version 1.0, some of the output is suppressed when constraints are in force.

pick `Exit I-NoLLS` to close down the program.

5. Applications

We developed I-NoLLS principally using real examples from molecular physics. These examples are described in Section 5.2 below. However, the complexity of the property calculation codes used in the real-world examples is such that it is not feasible to supply the whole suite of programs to Computer Physics Communications. We have therefore supplied code for a very small test problem, which nevertheless allows the principal features of I-NoLLS to be demonstrated. However, it should be remembered that I-NoLLS was developed specifically to handle cases where the property calculation is very expensive, and this particular feature is *not* shared by the test problem.

5.1. Example supplied

The test data file and CALCYJ subroutine supplied have been prepared for the example model function known as Rosenbrock's sum of squares, which has been extensively used for testing the convergence of least-squares methods [40,4]. It is defined as

$$\chi^2 = d_1^2 + d_2^2, \tag{23}$$

where

$$d_1 = 1 - p_1 \tag{24}$$

and

$$d_2 = 10(p_2 - p_1^2). \tag{25}$$

Here d_1 and d_2 correspond to the (observed-calculated) residuals in a least-squares problem. Since $m = n$, the fitting program is actually solving a nonlinear system of equations. This compact model function is adequate to illustrate the problems associated with severe nonlinearity and highly correlated parameters.

The quantity $\chi^2(p_1, p_2)$ is plotted as a contour diagram in Figs. 3 and 4. The minimum is at $(1, 1)$ and the correlation between p_1 and p_2 is 0.9988 at this point. Fig. 3 illustrates a least-squares fit using the automatic Levenberg-Marquardt (LM) algorithm described by Fletcher [40]. The starting point is $(-1.5, 1.5)$ with the LM parameter λ initially set to

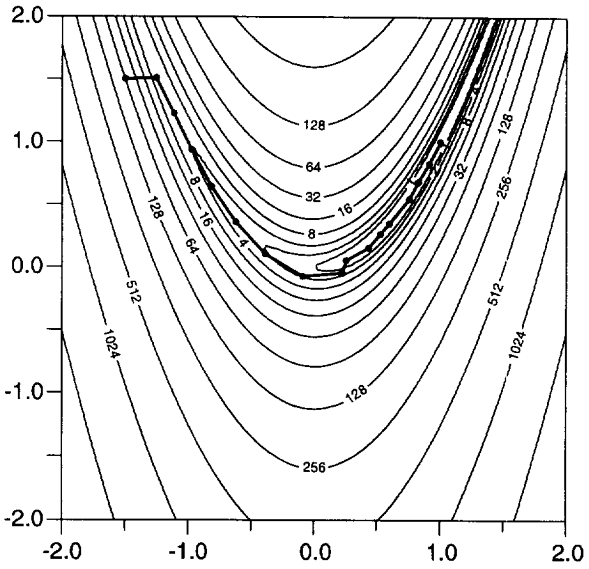

Fig. 3. Minimisation path for Rosenbrock's sum of squares using Fletcher's automatic Levenberg-Marquardt algorithm [40].

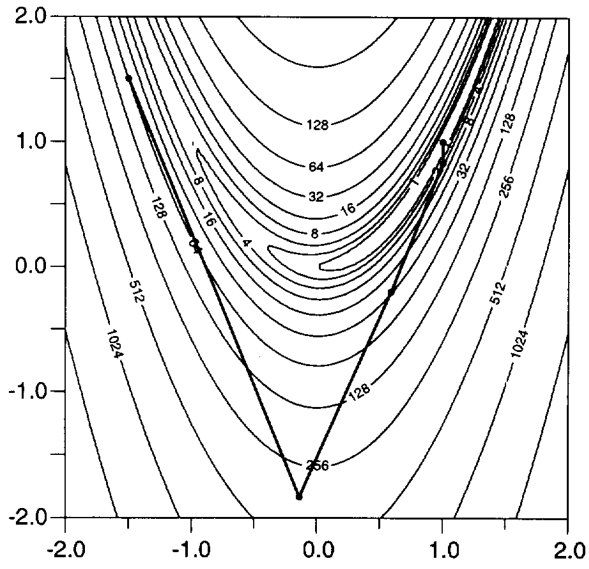

Fig. 4. Minimisation path for Rosenbrock's sum of squares using Fletcher's automatic Levenberg-Marquardt algorithm [40] with an initial step allowed to increase χ^2.

3.161. This latter value is 10 times the lower of the two singular values, and is large enough to ensure that the first step is 'downhill'. The algorithm follows the valley slowly along its curved path, and 16 iterations are required to get to within a distance from the solution (in parameter space) of 0.01. The length of each step is effectively restricted by the criterion that a step is considered acceptable only if it results in a reduction in χ^2.

In our work on fitting molecular potential energy surfaces we have frequently encountered analogous cases where an automatic algorithm makes very slow progress along a curved valley in the sum of squares. In such a situation it is often advantageous for the user to intervene and force the program to take one long step that actually *increases* χ^2. This gets the fit 'around the corner' of the valley and subsequent convergence of the fit is much more rapid. This behaviour is illustrated for Rosenbrock's function in Fig. 4. For the same starting conditions as before, λ is set initially to 0.3161 (equal to the lower singular value) and the fit is allowed to take the first, apparently unprofitable step before proceeding using Fletcher's automatic algorithm as above. In this case only 3 further iterations are required for convergence.

5.2. Interface with codes for molecular properties

As part of the development process, we have applied I-NoLLS to a number of real problems in chemical physics. Each application prompted improvements in the user interface or revealed the desirability of additional features and capabilities.

I-NoLLS has been interfaced with a number of codes for calculating molecular properties from potential energy surfaces:

- the BOUND code [11], for calculating spectroscopic properties of Van der Waals complexes;
- the TRIATOM [12] and DVR3D [13] codes, for calculating spectroscopic properties of chemically bound species;
- the MOLSCAT code [10], for calculating cross sections for inelastic molecular collisions;
- the VIRIAL code, for calculating second virial coefficients of gas mixtures.

In our pilot applications, the derivatives of the molecular properties required for the Jacobian matrices were calculated by *finite differences*. For an m-parameter problem this requires $m + 1$ evaluations of the set of properties for single-sided derivatives, or $2m + 1$ evaluations for two-sided derivatives. Under these circumstances, calculation of the super-Jacobian may always be parallelised. In practice each of the $m + 1$ subcalculations may be further subdivided according to molecular symmetry or quantum-mechanical considerations.

We have used the I-NoLLS/BOUND/MOLSCAT/VIRIAL code mostly on a cluster of 18 IBM RS/6000 workstations at Durham, using PVM (Parallel Virtual Machine) [43] library calls to pass the current values of the model parameters to the specialist dynamical codes and receive back the corresponding molecular properties. The specialised version of the subroutine CALCYJ used for these applications (not supplied to Computer Physics Communications) implements an efficient task-farming strategy aimed at achieving optimum load balancing between processors. Since only CALCYJ needs to be interfaced with the codes for calculating properties, it is straightforward to incorporate new versions of the codes, or codes for new properties (or even completely different applications).

We have also implemented a version of I-NoLLS/BOUND using CONDOR-PVM [44] on a large cluster of general-purpose Hewlett-Packard workstations in Durham. The CONDOR-PVM system allows the expensive dynamical calculations to use spare computing power on the workstations, while not interfering with interactive use.

5.3. Initial applications in molecular physics

I-NoLLS has been used to determine interaction potentials for He–Ar$^+$ [45], He–Kr$^+$ [46], He–HCN [47] and Ar–CO_2 [24]. Other applications are in progress.

The first application, to the He–Ar$^+$ ion [45], was motivated by measurements of the microwave spectrum of He–Ar$^+$ by Carrington et al. The spectra correspond to ions in vibration–rotation levels very near dissociation to form the $^2P_{3/2}$ and $^2P_{1/2}$ states of Ar$^+$. For these levels, the Born–Oppenheimer approximation has completely broken down, and coupled channel calculations are essential to reproduce the observed spectra accurately. We used the I-NoLLS/BOUND code to fit a potential energy surface (equivalent to a set of three potential curves and the couplings between them) to the microwave spectra and earlier results from ultraviolet spectroscopy. The final fit involved 88 data points and 7 free potential parameters [45]. This first real-world application was used to guide development of the I-NoLLS code, including its user interface and its interface with BOUND.

Carrington et al. subsequently measured analogous microwave spectra of He–Kr$^+$. These proved much more difficult to assign than those for He–Ar$^+$. However, with the help of I-NoLLS, we were able to carry out least-squares fits for a variety of tentative assignments of a subset of the lines until we found one that succeeded in explaining the remaining lines. This "bootstrap" approach eventually led to a near-complete assignment of the observed lines, and allowed us to obtain a full potential energy surface for He–Kr$^+$ [46]. This would have been a quite impossible task without the I-NoLLS program.

I-NoLLS has also been used to determine intermolecular potentials by fitting to spectroscopic data for the He–HCN and Ar–CO_2 Van der Waals complexes. The Ar–CO_2 application [24] included simultaneous fitting to second virial coefficients as well as spectroscopic data.

I-NoLLS is currently being applied to determining the intermolecular potential of Ar–H_2O by simultaneous fitting to the spectra of Van der Waals complexes (BOUND) and inelastic scattering cross sections (MOLSCAT).

I-NoLLS has also proved valuable in fitting to ab initio points for a variety of systems, including Ar_2–HF.

I-NoLLS can also be used in an "off-line" mode, in which the computationally expensive property calculations are performed remotely, for example on a parallel or vector supercomputer. In these circumstances it is most convenient for I-NoLLS to communicate with the property calculation codes via disc files rather than PVM calls. It would also be possible to implement this mode of operation using remote procedure calls.

6. Conclusions

The I-NoLLS program is a new interactive tool for difficult least-squares fitting problems. The program is valuable when conventional automated least-squares procedures fail, either because they take too much time carrying out property calculations at an excessive number of points in parameter space, or because the automated procedure strays into unphysical regions of parameter space. The I-NoLLS program allows the user to guide the progress of the fit by applying physical intuition, and to avoid parameter steps that appear unprofitable or unphysical. Least-squares steps can be tested before they are finally accepted, allowing the fit to recover from forays into unphysical regions of parameter space.

The I-NoLLS program has been interfaced to several packages for calculating molecular properties, and has been used for the determination of interatomic and intermolecular potential energy surfaces from spectroscopic and other data. However, I-NoLLS is specifically written to be highly modular, and can easily be interfaced with other codes for calculating different properties. It may thus be applied in a wide variety of optimisation problems in different fields.

Acknowledgements

This work was funded by the Science and Engineering Research Council as the "flagship" project of CCP6 (the Collaborative Computational Project on Heavy Particle Dynamics) for 1993–96. We are grateful to Jonathan Tennyson and Ian Mills for valuable discussions on the least-squares fitting problems encountered in fitting intramolecular potential energy surfaces, and to Keith Atkins and Andreas Ernesti for suggesting a number of improvements to the program. We are also grateful to Lydia Heck for assistance with PVM and to Nigel Starling for work on the CONDOR-PVM implementation. JMH thanks the Nuffield Foundation for the award of a one-year Science Research Fellowship in 1993–94. MML thanks the Royal Society for the award of a one-year Research Grant. Development work was carried out on a cluster of IBM RS/6000 workstations belonging to the atomic and molecular physics groups at Durham and Newcastle Universities and purchased with funds from the SERC Computational Science Initiative.

References

[1] L. Hedberg and I.M. Mills, J. Mol. Spectrosc. 160 (1993) 117.
[2] J.E. Dennis, in: Nonlinear Optimisation 1981, M.J.D. Powell, ed. (Academic Press, London, 1982).
[3] P.R. Bevington, Data Reduction and Error Analysis for the Physical Sciences (McGraw-Hill, London, 1969).

[4] P.E. Gill, W. Murray and M.H. Wright, Practical Optimization (Academic Press, London, 1981).
[5] I.M. Mills, in: Recent Experimental and Computational Advances in Molecular Spectroscopy, R. Fausto, ed. (Kluwer, London, 1993).
[6] J. Tennyson, Comput. Phys. Rep. 4 (1986) 1.
[7] Z. Bačić and J.C. Light, Ann. Rev. Phys. Chem. 40 (1989) 469.
[8] J. Tennyson, S. Miller and J.R. Henderson, in: Methods in Computational Chemistry, Vol. 5, S. Wilson, ed. (Plenum, New York, 1991).
[9] J.M. Hutson, Comput. Phys. Commun. 84 (1994) 1.
[10] J.M. Hutson and S. Green, MOLSCAT computer program, Version 14, distributed by Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council (1994).
[11] J.M. Hutson, BOUND computer program, Version 5, distributed by Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council (1993).
[12] J. Tennyson, S. Miller and C.R. Le Sueur, Comput. Phys. Commut. 75 (1993) 339.
[13] J.R. Henderson, C.R. Le Sueur and J. Tennyson, Comput. Phys. Commun. 75 (1993) 379.
[14] D.J. Nesbitt, M.S. Child and D.C. Clary, J. Chem. Phys. 90 (1989) 4855.
[15] T.S. Ho and H. Rabitz, J. Phys. Chem. 97 (1993) 13447.
[16] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Numerical Recipes (Cambridge University Press, Cambridge, 1986).
[17] D.L. Albritton, A.L. Schmeltekopf and R.N. Zare, Least-squares fitting of spectroscopic data, in: Molecular Spectroscopy: Modern Research, Vol. II, K.N. Rao, ed. (Academic Press, London, 1976).
[18] M.M. Law, J.M. Hutson and A. Ernesti, Fitting Molecular Potential Energy Surfaces, ISBN 0-9522736-0-8 Collaborative Computational Project No. 6 of the UK Engineering and Physical Sciences Research Council, Daresbury (1993).
[19] R.J. Le Roy and J.M. Hutson, J. Chem. Phys. 86 (1987) 837.
[20] J.M. Hutson, J. Chem. Phys. 96 (1992) 6752.
[21] J.M. Hutson, J. Phys. Chem. 96 (1992) 4237.
[22] S.C. Carter, I.M. Mills and N.C. Handy, J. Chem. Phys. 99 (1993) 4379.
[23] E.J. Bohac, M.D. Marshall and R.E. Miller, J. Chem. Phys. 97 (1992) 4890.
[24] J.M. Hutson, A. Ernesti, M.M. Law, C.F. Roche and R.J. Wheatley, J. Chem. Phys. 105 (1996) 9130.
[25] M.-L. Dubernet and J.M. Hutson, J. Chem. Phys. 99 (1993) 7477.
[26] U. Schnupf, J.M. Bowman and M.C. Heaven, Chem. Phys. Lett. 189 (1992) 487.
[27] R.C. Cohen and R.J. Saykally, J. Phys. Chem. 94 (1990) 7991.
[28] R.C. Cohen and R.J. Saykally, J. Chem. Phys. 98 (1993) 6007.
[29] C.A. Schmuttenmaer, R.C. Cohen and R.J. Saykally, J. Chem. Phys. 101 (1994) 146.
[30] M. Quack and M. Suhm, J. Chem. Phys. 95 (1991) 28.
[31] M.J. Elrod and R.J. Saykally, J. Chem. Phys. 103 (1995) 933.
[32] M.M. Law, J.L. Duncan and I.M. Mills, J. Mol. Struct. 260 (1992) 323.
[33] K. Rasmussen, S.B. Engelsen, J. Fabricius and B. Rasmussen, in: Recent Experimental and Computational Advances in Molecular Spectroscopy, R. Fausto, ed. (Kluwer, London, 1993).
[34] S.B. Engelsen, J. Fabricius and K. Rasmussen, Comput. & Chem. 18 (1994) 397.
[35] K. Levenberg, Quart. Appl. Math. 2 (1944) 164.
[36] D. Marquardt, SIAM J. Appl. Math. 11 (1963) 431.
[37] C.L. Lawson and R.J. Hanson, Solving Least Squares Problems (Prentice-Hall, New Jersey, 1974).
[38] J.E. Dennis, in: The State of the Art in Numerical Analysis, D. Jacobs, ed. (Academic Press, London, 1977).
[39] A.F. Krupnov and A.V. Burenin, in: Molecular Spectroscopy: Modern Research, Vol. II, K.N. Rao, ed. (Academic Press, London, 1976).
[40] R. Fletcher, Atomic Energy Research Establishment, Harwell Report R6799 (1971).
[41] L.S. Bartell, D.J. Romenesko and T.C. Wong, in: Molecular Structure by Diffraction Methods, Vol. 3, Specialist Periodical Reports (The Chemical Society, London, 1975) p. 72.
[42] Application Visualisation System computer code, Release 5.0, distributed by Advanced Visual Systems Inc., USA (1993).
[43] Parallel Virtual Machine computer code, Version 3.1, distributed by Oak Ridge National Laboratory, USA (1993).
[44] J. Pruyne and M. Livny, J. Future Generations of Computer Systems 12 (1996) 67.
[45] A. Carrington, C.A. Leach, A.J. Marr, A.M. Shaw, M.R. Viant, J.M. Hutson and M.M. Law, J. Chem. Phys. 102 (1995) 2379.
[46] A. Carrington, C.H. Pyne, A.M. Shaw, S.M. Taylor, J.M. Hutson and M.M. Law, J. Chem. Phys. 105 (1996) 8602.
[47] K.M. Atkins and J.M. Hutson, J. Chem. Phys. 105 (1996) 440.
[48] E. Anderson, Z. Bai, C. Bischof, J.W. Demmell, J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. C. Sorenson, LAPACK Users' Guide (SIAM, Philadephia, 1992).
[49] J.J. Dongarra, J. Du Croz, S. Hammarling and I. Duff, ACM Trans. Math. Softw. 16 (1990) 1.